

前端面试江湖

李红米 编著

電子工業出版社·

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书融合了 Web 前端面试题和主流开发技术,涵盖了 JavaScript、HTML5+CSS3、DIV+CSS、jQuery 等几大前端主流技术,结构清晰,层次明了。从知识点来说,基本上覆盖了近几年这个行业都会涉及的前端面试题;从实战经验来说,可以提高读者独立思考的能力,增加读者的实际应用技能。另外,本书的王牌“17 助力”,更能助读者在面试中的一臂之力。走进《前端面试江湖》不再为面试而苦恼!

本书适合从事前端工作或即将从事前端工作的读者阅读。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

前端面试江湖 / 李红米编著. —北京:电子工业出版社, 2016.5

ISBN 978-7-121-28507-3

I . ①前… II . ①李… III . ① JAVA 语言—程序设计 ②超文本标记语言—程序设计 ③网页制作工具 IV . ① TP312 ② TP393.092

中国版本图书馆 CIP 数据核字(2016)第 066169 号

策划编辑:陈晓猛

责任编辑:张 玲

印 刷:三河市双峰印刷装订有限公司

装 订:三河市双峰印刷装订有限公司

出版发行:电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本:720×1 000 1/16 印张:23.25 字数:446.4 千字

版 次:2016 年 5 月第 1 版

印 次:2016 年 5 月第 1 次印刷

定 价:69.00 元

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010) 88254888。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010) 88258888。

前言

互联网技术日新月异。2005 年以后，互联网进入 Web 2.0 时代，各种类似桌面软件的 Web 应用大量涌现，网站的前端由此发生了翻天覆地的变化。网页不再只是承载单一的文字和图片，各种丰富媒体让网页的内容更加生动，网页上软件化的交互形式为用户提供了更好的使用体验，这些都是基于前端技术实现的。

Web 前端开发是一项很特殊的工作，涵盖的知识面非常广，它既有具体的技术，又有抽象的理念。简单地说，它的主要职能就是把网站的界面更好地呈现给用户。随着前端开发涉及的领域愈发的广泛，各大企业对招聘的 Web 前端开发工程师岗位提出了更多的需求，对应聘者面试内容的知识面也不断拓宽，这就使很多应聘者卡在了面试这道坎儿上。

本书系统地划分了近几年来 Web 前端的主流模块和常用的框架、库等，整理了大量企业面试题，并附有资深前端工程师的详细讲解，独具特色的笔风让读者在学习的同时，也不会觉得枯燥。

为何写作本书

前端开发是个非常新的职业，对一些规范和最佳实践的研究都处于探索阶段，总有新的灵感和技术不时闪现出来，所以对前端开发工程师的能力要求越来越高。由于前端开发人员的需求越来越大，更多的人愿意学前端，做前端，所以前端开发的就业前景越来越广，竞争也越来越激烈。一位好的 Web 前端开发工程师在知识体系上既要有深度，又要有广度，所以很多大公司即使出高薪也很难招聘到理想的前端开发工程师。然而公司对应聘者的初次了解就在面试中，可见这“第一印象”很重要。

为了有效地应对面试，“面经”这个新兴概念应运而生。笔者在当初找工作阶段也从“面经”中获益匪浅。但“面经”中的分类不是很清晰，具有针对性的试题和经验也很难找到。为了方便后来者，笔者花费大量时间对企业真实面试题进行了改编，并从自身专业出发，着重关注前端程序员面试，并从中精选

出若干具有代表性的技术类面试题进行深入剖析，希望能给前端面试人员提供更多参考。

本书特色

面试实战

本书面试题全都来源于企业面试。近几年来，关于 Web 前端开发职位的要求越来越高，致使大多数人在面试时出师不利。本书所包含的面试题中，涵盖了前端最新的几大主流技术，以及近几年前端所涉及的大多数知识。

深入解析

本书用相当多的篇幅重点剖析试题，并配有 Web 前端资深工程师的解答，其中不乏前端应用中的方法与技巧。通过解析模拟面试题，能有效帮助读者掌握 Web 前端技术，使读者对前端面试更有把握。

笔风特色

本书名为《前端面试江湖》，笔者在编著此书时，将前端当成一个江湖，江湖分支有 JavaScript、HTML5+CSS3、DIV+CSS、jQuery 等几大前端主流技术，书中还汇聚了从筑基修炼到终极飞升的面试题。相信读者会在阅读的过程中感觉到做 Web 前端的不易，但我们也相信，你们会有斗志继续攻克前端，并在学习与拓宽前端的路上越走越远。

本书内容结构

本书分为六篇，共 19 章，具体结构划分如下。

第 1～4 篇：试题演练讲解部分，共 16 章。这部分主要介绍 JavaScript、HTML5+CSS3、DIV+CSS、jQuery 等几大前端主流技术。其中包含初级面试题、中级面试题、高级面试题、终极面试题。

第 5 篇：本篇内容是前端开发“17 助力”，记录一些有关前端岗位的综合面试题，以及在面试过程中，除了我们在前面提出的几大主流技术外，大多数企业面试题会涉及的知识点，提升读者对前端知识的认识，并且本篇还有考验程序员的逻辑思维的试题。

第 6 篇：本篇共两章，主要内容为综合测试题和人力资源面试相关的技巧，为读者提供模拟面试题和人力资源常问的问题，为您的面试保驾护航。

本书面向的读者

- 具有一定的 Web 前端基础的求职者
- 具有计算机编程基础或从计算机后端转前端的求职者
- 各大院校学习计算机课程，并今后想从事 Web 前端工作的求职者
- Web 前端开发在职人员

关于我们

在本书的写作过程中，策划编辑陈晓猛给予了很大的帮助和支持，在此表示感谢。

另外，书中教给读者的方法是次要的，重要的是这本书可以增加读者的思维能力，能够让人做到举一反三。

由于时间有限，书中可能会出现一些错误或读者不理解的地方，读者可以发送邮件到 kf@qdjhu.com，也可以登录前端江湖的官网（www.qdjhu.com），在我们的问答模块上提出您的反馈意见。我们谨以最真诚的心，希望能与读者共同成长。

“只会在水泥上走路的人，永远不会留下深深的脚印”，亲爱的读者朋友，希望在前端的道路上我们一起走过！

李红米
2016.2.1

目 录

第 1 篇 烽烟起，剑在手——JavaScript 永不休

第 1 章 未入江湖，先定三分 [JavaScript 初级面试题].....	2
1.1 面试从这里开始（关于 JavaScript）.....	2
1.2 这题很简单（字符串）.....	3
1.3 我面试，我快乐（变量）.....	5
1.4 平平淡淡搞定面试（数据类型）.....	7
1.5 哎呦，这题不错哦（window 的属性和方法）.....	8
1.6 让你魂牵梦绕的题（元素）.....	12
1.7 “神马”题都是浮云（className）.....	15
1.8 看完这题，花儿都笑了（保留字和关键字）.....	15
1.9 我懂的题，你懂吗（循环语句）.....	16
第 2 章 能文能武，身怀绝技 [JavaScript 中级面试题].....	17
2.1 小考题，大味道（表达式）.....	17
2.2 让人叹为观止的考题（String 对象）.....	18
2.3 让人绞尽脑汁的考题（Math 对象）.....	21
2.4 令人陶醉的考题（数组）.....	23
2.5 令人难过的考题（函数）.....	34
2.6 令人抑郁的考题（对象）.....	40
2.7 夜以继日难以攻克的考题（定时器）.....	50
2.8 学无止境（日期和时间）.....	52
2.9 主流技术的“最爱”（typeof、instanceof）.....	54
第 3 章 进得武林，入得四方 [JavaScript 高级面试题].....	56
3.1 轻松解决缠绕你的考题（this）.....	56
3.2 细心可以拿满分的题（事件）.....	58
3.3 不可忽视的小漏洞（表单、文本框）.....	68

3.4 “照镜子”看题（对称数）	77
3.5 让你坚定不移看下去（JavaScript 客户端检测）	77
3.6 过了这题，公司随便挑（排序）	78
3.7 看了这题，收获多多（call、apply）	79
3.8 看懂必高薪的面试题（继承和多态）	80
3.9 大型企业面试必考（charAt()、indexOf()）	82
3.10 五年前端，三年必考（substr、substring）	82
3.11 这题如此而已，我会为你加油（iframe）	82
第4章 一手遮天，大名远扬 [JavaScript 终极面试题]	84
4.1 让人暴走的考题（Ajax）	84
4.2 面对这些考题，除了崩溃我不知道还能说什么 （XMLHttpRequest 对象）	90
4.3 感慨“时间太瘦”的考题（关于继承）	92
4.4 高级前端必考试题（闭包）	97
4.5 值得思考和深钻的考题（video）	99
4.6 这些年，一直“陪伴我”的考题（url 参数）	100
4.7 小心陷阱！总是被坑的考题（JavaScript 模仿块级作用域）	101
4.8 错误率最高的考题（正则表达式）	102
4.9 前端最新技术考题（转换大写）	103
4.10 总有一种题，叫看起来都对（JSON）	103
4.11 最难回答的考题（事件委托）	107
4.12 高智商考题（事件流）	109
4.13 前端压轴考题（错误处理与调试）	115
4.14 你值得拥有的考题（Cookie）	115
综合提升	116

第2篇 驾骏马，拉长弓——HTML5+CSS3 我独行

第5章 博学多才，雄韬伟略 [HTML5+CSS3 初级面试题]	128
5.1 做了这些，不再是菜鸟（关于 HTML5）	128
5.2 真本事，更自信（HTML5 语法）	129
5.3 这些题总能温暖你（HTML5+CSS3 新增属性）	131

5.4	最实用的题（HTML5 与 XML）	132
5.5	KO 这些题，前端岗位不是梦（HTML5 结构）	133
5.6	前端好考题（HTML5 布局）	134
5.7	这些题，让你赢在起跑线上（关于 CSS3）	135
5.8	领先别人一步（CSS3 选择器）	135
5.9	从最陌生到最熟悉的题（切图）	136
5.10	做好当下（定位相关）	136
第 6 章	不鸣则已，一鸣惊人 [HTML5+CSS3 中级面试题]	138
6.1	没有你们，我会不安（HTML5 音频与视频）	138
6.2	终是拨开云雾见月明（弹性盒布局）	139
6.3	心在天上，题在手上（HTML5 常见问题）	140
6.4	多几分钟的准备，少几小时的麻烦（HTML 元素）	140
6.5	经久不衰的考题（代码纠错）	141
6.6	比上不足，比下有余（代码优化）	142
6.7	非常可乐，非常选择（上传）	143
6.8	爱上面试的感觉（文本）	144
6.9	你想摆谱，先干掉我（字体）	146
6.10	前端深处考题（边框背景）	147
6.11	有这些，更自信（多列布局）	148
6.12	总有些考题念念不忘（多列显示样式）	149
6.13	深入每道题的世界（盒布局）	150
第 7 章	运筹帷幄，决胜千里 [HTML5+CSS3 高级面试题]	151
7.1	就这些，永不过期（模式）	151
7.2	这些题，让你前端技艺更高一筹（HTML5 页面）	152
7.3	考题中的钉子户（Canvas 的使用）	153
7.4	点面试，闯全关（媒体查询）	154
7.5	面试一大坎儿（浏览器缓存与本地储存）	154
7.6	这些让你更强大（媒体调用标签）	155
7.7	脑若一动，题就千行（HTML、CSS 综合）	156
7.8	有了我就知足吧（兼容问题）	157
7.9	一直在寻找，直到遇见你（响应式布局）	157

7.10	一直在寻觅的考题（关于浏览器）	158
7.11	无法轻描淡写的考题（PC、移动）	159
7.12	最难懂的题给真心的你（HTML5 效果）	160
7.13	从此，面试不重来（控件相关）	161
第 8 章	见多识广，独霸一方 [HTML5+CSS3 终极面试题]	163
8.1	这些题必须认真对待（HTML5 应用程序缓存）	163
8.2	这些题“包罗万象”（HTML5 常见 API）	164
8.3	再深的题海，也能乘风破浪（HTML5 数据存储）	164
8.4	这些题让你相信能，就能！（HTML5 编辑 API）	165
8.5	搞清楚这些让你“屌炸天”（CSS 动画）	165
8.6	前端面试独家宝贝（cache 机制）	167
8.7	前端大牛的看家本事（workers 多线程处理）	168
8.8	一入考题深似海，从此面试是浮云（Geolocation 地理位置）	169
8.9	前端大“虾”必考题（编码问题）	170
	综合提升	170

第 3 篇 箭在弦，不回头——DIV+CSS 向前冲

第 9 章	夯实基础，厚积薄发 [DIV+CSS 初级面试题]	180
9.1	有一种题叫边做边流泪（浮动）	180
9.2	想登上理想的高峰吗？那就来吧（块级元素）	182
9.3	时间很短，面试赶紧（行内元素）	182
9.4	面试就像半杯水，你能看到什么（CSS 图片）	183
9.5	面试是一场旅行，正在进行时（CSS 引入）	184
9.6	面对考题不曾退缩（CSS 概述）	184
9.7	面试者就像蒲公英，看似自由，却身不由己（CSS 选择器）	186
9.8	绝地逢生之“路”（CSS 规范）	187
9.9	能磨炼薄弱意志的考题（HTML 结构）	188
9.10	看的越少，失去的越多（隐藏 DOM 元素）	190
9.11	喜欢前进，看的题就越来越多（CSS 文字样式）	191
第 10 章	百折不挠，历经磨难 [DIV+CSS 中级面试题]	192
10.1	有一种题做起来很崩溃（inline-block 特性）	192

10.2	快到题里来（布局）	192
10.3	思想太满，就会学不来（CSS 属性）	193
10.4	面试失败十次，找第十一次坚持的借口（清除浮动与闭合浮动）	193
10.5	多项选择，任你选择（CSS 定义标签）	194
10.6	不满足昨天的难度（简化 CSS 代码）	195
10.7	华丽的跌倒，胜过无谓的徘徊（CSS 设置图片效果）	196
10.8	缘分是一本书，翻得不经意会错过（CSS 设置表格）	197
10.9	不要在错的题上犹豫不决（背景图片）	197
10.10	面试是一把锁，你拿对钥匙了吗（CSS 与表单）	198
10.11	有思维才是王牌（CSS 定位）	199
10.12	提前进入，囤积自信（DIV 布局）	200
10.13	拥有别人没有的（盒子布局）	202
10.14	既然无处可逃，不如帅气迎接（CSS 排版）	203
10.15	“限量版”的题（CSS 注释）	204
10.16	值得分享的考题（iframe）	205
第 11 章	出类拔萃，终成大器 [DIV+CSS 高级面试题]	206
11.1	程序“猿”你懂了吗（DOCTYPE）	206
11.2	面试其实很简单，就看你了（CSS Hack）	207
11.3	别让面试，输给了心情（悬浮效果）	207
11.4	面试不要瞎忙，不经意的才是最好的（CSS 优先级）	207
11.5	我不知将去何方，但我已在路上（定位）	208
11.6	面试如一道弧线，却能摆平一切（页面布局）	210
11.7	程序员，我不知道你心里是怎么想的（CSS 字体）	211
11.8	面试时谁没有耐心，谁就没有智慧（CSS 表格）	211
11.9	面试，你紧张了吗（CSS 内外边距）	212
11.10	不能白看，看完必过（CSS 文本）	212
11.11	面试如同千军万马，更是使人心惊胆战（CSS 链接）	213
第 12 章	炉火纯青，大杀四方 [DIV+CSS 终极面试题]	216
12.1	面试就像自行车，说得简单，其实还要靠自己（三层构成）	216
12.2	面试就算终有一散，也别辜负相遇（CSS Sprites）	216
12.3	面试短短的话语，却包含万千（CSS 中 a 的伪类）	217

12.4 面试要勇敢，前方的路很长（CSS 浏览器兼容）	218
12.5 程序“猿”看完就乐了（CSS 水平对齐）	222
12.6 “面试”是一件多么美的事（CSS 浮动）	222
12.7 面试是不可缺少的美好亮点（CSS 优势）	224
12.8 你可知道面试的重要（IE6 常见问题）	224
综合提升	226

第4篇 江湖路，无尽头——jQuery 任逍遥

第13章 刻苦学艺，心无旁骛 [jQuery 初级面试题]	232
13.1 要想简单，那就简单（选择器）	232
13.2 面试总会有不期而遇的温暖（属性）	235
13.3 不要为面试而烦恼（绑定事件）	237
13.4 不要害怕面试，因为你需要（表单）	238
13.5 说多了都是眼泪，还是来点实在的吧（文档处理）	239
13.6 最怕的东西，最应该去突破（筛选）	239
第14章 学贯古今，中流砥柱 [jQuery 中级面试题]	240
14.1 你还在“泡”招聘，“奔”面试吗（DOM 加载）	240
14.2 你能让面试官惊呆了吗（移动端事件）	240
14.3 面试是一张网，你收获了吗（取 HTML、文本的值）	241
14.4 每天超越自己一点点（事件）	241
第15章 出神入化，学贯古今 [jQuery 高级面试题]	243
15.1 做小题，成大事（read、onload 的区别）	243
15.2 让愤怒多些实力（效果）	243
第16章 英姿勃发，独当一面 [jQuery 终极面试题]	246
16.1 困难的考题能让你看到更多的风景（get 和 post）	246
16.2 想知道你能力的边界在哪吗（优化）	246
16.3 其实成功一直在你的旁边（Ajax）	247
16.4 放手做，勇敢错（jQuery、DOM 对象）	248
16.5 断了退路，才有出路（\$.getScript() 和 \$.getJSON()）	251
综合提升	252

第5篇 清风落，江湖生——「17助力」泯恩仇

第17章 万事俱备，只欠东风 [“17助力”，助你一臂之力]	258
17.1 “助力1”：浏览器和兼容差异	258
17.2 “助力2”：前端优化	264
17.3 “助力3”：开发者工具	267
17.4 “助力4”：JS库和框架	269
17.5 “助力5”：cookie	276
17.6 “助力6”：超级素数	277
17.7 “助力7”：主流技术	277
17.8 “助力8”：进制转化	278
17.9 “助力9”：追加字符串	279
17.10 “助力10”：模块模式	279
17.11 “助力11”：效果题	281
17.12 “助力12”：跨域问题	285
17.13 “助力13”：前端交谈	285
17.14 “助力14”：综合考察	298
17.15 “助力15”：项目问题	298
17.16 “助力16”：Flash	302
17.17 “助力17”：逻辑题	303

第6篇 功成时，把酒笑——综合测试莫言愁

第18章 前端开发面试题	310
18.1 前端面试模拟试题一	310
一、技术题	310
二、效果题	349
18.2 前端面试模拟试题二	351
第19章 人资问题	356

第1篇

烽烟起 ——JavaScript永不休

，剑在手

JavaScript (JS, JavaScript 的简写形式), 一种直译式脚本语言, 为浏览器的一部分, 最早是在 HTML 网页上用来给 HTML 网页增加动态功能的。本篇全面涵盖了 JavaScript 的各种初级、高级、主流的特性, 能够让我们更好地认识和接触 JavaScript 的强大功能。

第 1 章



未入江湖，先定三分 [JavaScript 初级面试题]

踏入江湖第一步，首先要有最基本的保命能力，JavaScript 中最基本的原理将带领你踏入江湖，领悟其中奥妙，即可称霸 JS 江湖。

1.1 面试从这里开始（关于 JavaScript）

第 1 道：关于 JavaScript 说法正确的是（ ）

- A. 它是面向对象的
- B. 它是基于对象的
- C. 它是面向过程的
- D. 以上说法都不正确

答案：B

解析：

JavaScript 是基于对象的、事件驱动的脚本程序设计语言。所谓对象是指任何事物都可以封装成类，而一个类中具体的某个事物就是这个类的对象；所谓事件是指为这些事件定义一些动作，当处理这些事件后就执行相关代码，进行一些操作。

第 2 道：下列选择中写法正确的有（ ）

- A. `<script language="javascript"></script>`
- B. `<javascript language="javascript"></javascript>`
- C. `<javascript script="javascript"></javascript>`
- D. `<script javascript></script>`

答案：A

解析：

`<script language="javascript">` 用来告诉浏览器，这是用 JavaScript 写的程序，需要调动相应的解释程序进行解释。

第 3 道：请选择对 JavaScript 理解有误的项（ ）（多选）

- A. Jscript 是 JavaScript 的简称
- B. JavaScript 是网景公司开发的一种 Java 脚本语言，其目的是为了简化 Java 的开发难度
- C. Firefox 和 IE 存在大量兼容性问题，其主要原因在于它们对 JavaScript

的支持不同

D. Ajax 技术一定要使用 JavaScript

答案：AB

解析：

JavaScript，简称 JS。它是通过嵌入或调入标准的 HTML 语言来实现的。Ajax 的全称是 Asynchronous JavaScript and XML，基于 XML 的异步 JavaScript。Firefox 和 IE 存在大量兼容性问题，主要原因在于它们对 Web 的标准支持不同。

1.2 这题很简单（字符串）

第 4 道：如何截取字符串 www.qdjhu.com 中的 qdjhu？

解析：

采用 substr 方法。

定义和用法：substr 方法用于返回一个从指定位置开始的指定长度的子字符串。

语法：stringObject.substr(start [, length])

参数描述：

- start 必需。它是所需的子字符串的起始位置。字符串中的第一个字符的索引为 0。
- length 可选。指在返回的子字符串中应包括的字符个数。

示例代码：

```
var siteurl = 'www.qdjhu.com'; // 前端江湖网址
var result = siteurl.substr(4,5);
```

第 5 道：判断字符串是否是这样组成的，第一个必须是字母，后面可以是字母、数字、下划线、总长度为 5 ~ 20。

解析：

采用正则的匹配方法。^{^[a-z]} 是匹配小写的 26 个字母，^[A-Z] 是匹配大写的 26 个字母，^{n} 表示匹配几位，^{\w} 表示任意的字母、数字、下划线。

注意：正则有大写小写之分。

代码如下：

```
var c = /^[a-zA-Z]{1}\w{4,19}/;
```

第 6 道：编写一个方法，求一个字符串的字符长度。

解析：

用 length 方法来获取，返回值为 Number。

代码如下：

```
function getLength(str){
```

```
        return str.length;
    }
    var result=getLength("www.qdjhu.com");
```

第 7 道：给你一个字符串 String="adadfdfseffqjdjhuserfefsefseetsdg", 要求找出里面的字符串 qdjhu, 使用 JavaScript 实现。

解析：

这里要用到 indexOf()。

返回字符 indexOf (string) 中字符串 string 在父串首次出现的位置, 从 0 开始! 没有返回 -1; 方便判断和截取字符串!

代码如下:

```
var str="adadfdfseffqjdjhuserfefsefseetsdg";
var search='qdjhu';
var start=str.indexOf(search);
var result=str.substring(start,start+search.length);
```

第 8 道：如何获取浏览器 URL 中查询字符串的参数？

解析：

```
function getQuery(name)
{
    var reg = new RegExp("(^|&)" + name + "=(^[&]*)(&|$)");
    var r = window.location.search.substr(1).match(reg);
    if (r!=null) return unescape(r[2]); return null;
}
```

第 9 道：如何实现一个删除字符串左边空白字符的方法？

解析：

去掉字符串左边的空格:

```
function leftTrim(str){
    return str.replace(/^\s*/g,""); //^符号表示从开头（即左边）进行匹配
}
```

第 10 道：JavaScript 的 typeof 都返回哪些数据类型？

解析：

undefined、boolean、string、number、object、function。

第 11 道：写出以下语句运算结果的语句。

A. typeof (null) B. typeof (undefined) C. typeof (NaN) D. typeof (NaN==undefined)

解析：

A. “object” B. “undefined” C. “number” D. “boolean”

第 12 道："5"+3 的结果是多少？为什么？

解析：

结果是 53。这时的“5”是一个字符串，“+”起的是连接的作用，不是相

加，所以结果是 53。

第 13 道：请自定义一个函数，实现字符串反转。

解析：

JavaScript 实现字符串反转主要是把字符串从末尾开始的每一个元素截取后，再重新组成一个新的字符串。代码如下：

```
function revStr(str){
    var tmpStr="";
    for(var l=str.length-1;l>=0;l--){
        tmpStr+=str.charAt(l);
    }
    return tmpStr;
}
var str="abcdeg";
console.log(revStr(str));
```

第 14 道：字符串操作主要有哪些？

解析：

- (1) 求字符串长。
- (2) 字符串赋值。
- (3) 连接字符串操作。
- (4) 求子串。
- (5) 字符串比较。
- (6) 子串定位。
- (7) 字符串插入。
- (8) 字符串删除。
- (9) 字符串替换。

1.3 我面试，我快乐（变量）

第 15 道：JavaScript 中变量声明有 var 和没 var 的区别。

解析：

JavaScript 中变量声明的作用域是以函数为单位的，在函数内部，有 var 和没 var 声明的变量是不一样的。有 var 声明的是局部变量，没 var 声明的是全局变量。在 JavaScript 的函数作用域内，声明的变量或内部函数在函数体内都是可见的。这意味着，函数在定义之前可能已经可用。函数定义有两种方式，一种是函数定义表达式，一种是函数声明语句。函数声明语句“被提前”到外部脚

本或外部函数作用域的顶部。所以，以这种方式声明的函数，可以被它定义之前出现的代码所调用。而函数定义表达式中，变量的声明被提前了，但是给变量的赋值是不会提前的。所以，以表达式方式定义的函数在函数定义之前无法调用。

在全局作用域内声明变量时，有 `var` 和没 `var` 是有区别的。使用 `var` 语句重复声明语句是合法且无害的。如果重复声明且带有赋值，那么就和一般的赋值语句没差别。如果尝试读取没有声明过的变量，JavaScript 会报错。

第 16 道：在 JavaScript 中，以下哪个变量名是非法的（ ）？

A. Name B. 9Name C. Name_a D. Name9

答案：B

解析：

第一个字符必须是一个 ASCII 字母（大小写均可），或一个下画线“_”。注意第一个字符不能是数字，后续的字符必须是字母、数字或下画线。

第 17 道：在 JavaScript 中，（ ）变量在函数外声明，并可从脚本的任意位置访问。

A. 局部 B. 全局 C. `typeof` D. `New`

答案：B

解析：

全局变量无论是在函数内部还是外部都可以访问，而局部变量只能在函数内部访问。

第 18 道：JavaScript 中如何检测一个变量是一个 `string` 类型？请写出函数实现。

解析：

```
var str = "hello world";
function isString(str) {
    if (typeof str == "string" || str.constructor == String) {
        return true;
    }else{
        return false;
    }
}
```

第 19 道：计算下面的变量值：

```
var a=(Math.PI++);
var b=(Math.PI++);
alert(a);
alert(b);
```

解析：

`Math.PI` 是圆周率数值。输出是 3.141592653589793，没有改变，说明系统对象的属性是只读的。

第 20 道：有一个字符串 abcd-ef-ghi，请用 JavaScript 将它处理成 ghi&ef&abcd。

解析：

```
var str='abcd-ef-ghi';
var arr1=str.split('-');
var result=arr1.reverse().join("&");
```

第 21 道：“undefined”变量和“undeclared”变量分别指什么？

解析：

根本的区别在于，undefined 是 JavaScript 语言类型，而 undeclared 却是一种 JavaScript 语法错误。

在 JavaScript 中，有两个表示“空”的值：undefined 和 null，其中比较有用的是 undefined。undefined 是一个值为 undefined 的类型。JavaScript 语言也定义了一个全局变量，它的值是 undefined，这个变量也被称为 undefined。但是这个变量不是一个常量，也不是一个关键字。这意味着它的值可以轻易被覆盖。为了避免可能对 undefined 值的改变，一个常用的技巧是使用一个传递到匿名包装器的额外参数。在调用时，这个参数不会获取任何值。

而 undeclared 则是一种语法错误，其实访问 undeclared 的变量并非会中断浏览器执行。在浏览器运行上下文中，undeclared 出来的变量简单可以认为没有 var a 这样定义变量。JavaScript 引擎执行的时候，由于无法找到其对应的上下文（scope），所以会简单地认为该变量是全局的变量，就是会把该变量定义到 window 中去。

第 22 道：JavaScript 的两种变量范围有什么不同？

解析：

全局变量：当前页面内有效。

局部变量：方法内有效。

1.4 平平淡淡搞定面试（数据类型）

第 23 道：列举 JavaScript 中的数据类型。

解析：

原始数据类型：

1. Undefined 类型

Undefined 类型只有一个值，就是 undefined。当声明变量未初始化时，该变量默认就是 undefined。如果函数没有返回值，也会显示 undefined。

2. Null 类型

Null 类型（空型）只有一个值，就是 null。

undefined 实际上是从 null 派生来的，所以显示 true。

undefined 是声明了变量但是未赋值，null 是找不到对象。

3. Boolean 类型

非 0 即真，0 可以看成 false。

4. Number 类型

Number 类型中的所有值都在最大值和最小值之间。如果非数字 NaN (Not a Number) 是一个特殊的值。判断是否是数字用 isNaN()。

5. String 类型

理论上 String 可以无限制存放 Unicode 字符。赋值时双引号和单引号都一样。

复合数据类型：

function 函数类型 [*]object 对象类型 ,object 本质是由一组无序的名值对组成的。

array 数组类型（它是一种特殊的 object 对象类型）检查一个变量的数据类型。

第 24 道：JavaScript 中基本数据类型有哪些？与 primitive 数据类型有哪些不同？

解析：

在 JavaScript 中有 4 种基本的数据类型：数值（整数和实数）、字符串型（用 "" 或 ' ' 括起来的字符或数值）、布尔型（用 true 或 false 表示）和空值。在 JavaScript 的基本类型中的数据可以是常量，也可以变量。

primitive 数据类型有：byte（字节）、short（短整型）、int（整型）、long（长整型）、float（浮点型）、double（双精度）、char（字符串）。

第 25 道：false==0 的结果是多少？

解析：

因为逻辑值真 (True) 和假 (False 被当作数值运算时，其值分别为 1 和 0。

所以 MAX(0,-1,True) 结果为 1，MIN(False,1,2) 结果为 0。

故 false==0 结果为 false。

1.5 哎哟，这题不错哦（window 的属性和方法）

第 26 道：如何实现 alert 中的换行？

解析：

alert 中实现换行的是 \n。代码如下：

```
<script language="JavaScript" type="text/javascript">
    alert("hello\nworld");
</script>
```

第 27 道：要动态改变层中内容可以使用的方法有（ ）（多选）

- A. innerHTML
- B. innerText
- C. 通过设置层的隐藏和显示来实现
- D. 通过设置层的样式属性的 display 属性

答案：AB

解析：

通过改变 DIV 的 innerHTML 属性值动态改变页面内容。这种情况适合动态显示的内容较少时使用，所以，当动态显示的内容（如“用户名”不能为空）占据一行时比较适合此种方法，使用 mydiv.innerHTML="html 代码" 来动态改变页面内容。

第 28 道：document.write 和 innerHTML 的区别。

解析：

write 是 document 对象的一个方法，是在页面里写内容，它会覆盖页面内容，是写死的；innerHTML 是 DOM 元素对象的一个属性，用于设置内容。

第 29 道：在 HTML 中，下面属于段落标签的是（ ）

- A. <html></html>
- B. <head></head>
- C. <body></body>
- D. <p></p>

答案：D

解析：

网页的内容需在 <html> 标签中，标题、字符格式、语言、兼容性、关键字、描述等信息显示在 <head> 标签中，而网页需展示的内容需嵌套在 <body> 标签中，段落文本则需在 <p> 标签中。

第 30 道：有如下代码，在不改写已有 body 部分代码情况下，编写 JavaScript 函数来实现单击 id 为 test 的列表中的 li 时，获取其内容，比如点第一个 li 时，alert 出“第一行”。

```
<body onload="getTestLi()">
    <ul id="test">
        <li>第一行</li>
        <li>第二行</li>
        <li>第三行</li>
    </ul>
</body>
```

解析:

```
function getTestLi(){
    var
        objUlLi=document.getEleme
        ntById('test').getElementsB
        yTagName('li');

    for(i=0,l=objUlLi.length;i
    <l;i++){
        objUlLi[i].onclick=functio
        n(){ alert(this.innerHTML)
        ; };
    }
}
```

第 31 道：JavaScript 中的三种弹出式消息提醒，分别为 alert、confirm、prompt，请简要阐述。

解析:

- (1) alert——弹出警告框。在文本里面加入“\n”就可以换行。
- (2) confirm——弹出确认框。会返回布尔值，通过这个值可以判断单击时是“确认”还是“取消”。true 表示单击了“确认”，false 表示单击了“取消”。
- (3) prompt——弹出输入框。单击“确认”返回输入框中的值，单击“取消”返回 null。

第 32 道：请实现鼠标单击页面中的任意标签，alert 该标签的名称（注意兼容性）。

解析:

```
<script type="text/JavaScript">
    document.onclick=function(e){
        var e=(e||event);
        var o=e["target"]||e["srcElement"];
        alert(o.tagName.toLowerCase());
    }
</script>
```

有些标签没有 click 事件，例如当标签是 object 时，就没有 click 事件。

第 33 道：下面的 JavaScript 代码段中，alert 的结果是多少？

```
var a=1;
function f(){
    alert(a);
    var a=2;
}
f();
```


解析：

函数 a 与变量 b 重名，后面的赋值语句会使前面的函数对象失效，而使 a 变成一个普通变量。

JavaScript 是预编译机制，会使函数定义被优先处理，即使改变语句顺序也不会改变输出结果，所以 alert 的结果是 1。

第 34 道：结合 `text` 这段结构，谈谈 innerHTML、outerHTML 之间的区别。

解析：

innerHTML 设置或获取位于对象起始和结束标签内的 HTML；

outerHTML 设置或获取对象及其内容的 HTML 形式。

第 35 道：关于 IE 的 window 对象表述正确的有（ ）（多选）

A. window.opener 属性本身就是指向 window 对象

B. window.reload() 方法可以用来刷新当前页面

C. window.location="a.html" 和 window.location.href="a.html" 的作用都是把当前页面替换成 a.html 页面

D. 定义了全局变量 g，可以用 window.g 的方式来存取该变量

答案：ACD

解析：

window.opener 是 JavaScript 中 window 的一个属性，它返回的是打开当前窗口的窗口对象。如果窗口 A 弹出一个窗口 B，那么在 B 中 window.opener 就是窗口对象 A；window.location.reload() 方法是用来刷新当前整个页面的。

第 36 道：单击一个按钮，如何刷新整个页面？

解析：

首先介绍两个方法的语法：

(1) reload 方法强迫浏览器刷新当前页面。

语法：location.reload([bForceGet])；

参数：bForceGet，可选参数，默认为 false，从客户端缓存里取当前页。如果为 true，则以 GET 方式，从服务端取最新的页面，相当于客户端单击“F5”（刷新）。

(2) replace 方法通过指定 URL 来替换当前缓存在历史里（客户端）的项目。因此，当使用 replace 方法之后，不能通过“前进”和“后退”来访问已经被替换的 URL。

语法：location.replace(URL)；

在实际应用中，当重新刷新页面时，我们通常使用 location.reload() 或者是 history.go(0)。因为这种做法就像是在客户端单击“F5”刷新页面，所以页面的 method="post" 的时候，会出现“网页过期”的提示。那是因为 Session 的安

全保护机制。可以想到，当调用 `location.reload()` 方法时，此时 `aspx` 页面在服务端内存里已经存在，因此必定是 `IsPostBack` 的。如果有这种应用，我们需要重新加载该页面，也就是说我们期望页面能够在服务端重新被创建，我们期望是 `Not IsPostBack` 的。这里，`location.replace()` 就可以完成此任务。被 `replace` 的页面每次都在服务端重新生成。你可以这么写：`location.replace(location.href)`。

1.6 让你魂牵梦绕的题（元素）

第 37 道：JavaScript 中获取某个元素有哪几种方式？

解析：

`document.getElementById()` 用于获得名为 ID 值的元素；

`document.myform.xxx` 按照层次结构来获取；

`document.getElementsByName()` 用于获得所有的名字相同的元素。

第 38 道：怎样创建、查找、删除 DOM 节点，如何修改 DOM 相应属性？

解析：

创建 DOM 节点：`var odiv = document.createElement('div');`

查找 DOM 节点：`getElementById()`；

删除 DOM 节点：`document.body.removeChild(oP)`。

第 39 道：怎么捕获节点（问的频率高）？解释一下 `getElementById()` 和 `getElementsByTagName()`。

解析：

`document.getElementById("节点 id");` // 根据节点 id 获取节点，返回节点对象

`document.getElementsByTagName("标签名");` // 根据标签名获取该标签表示的节点对象数组

`document.getElementsByName("name 属性值");` // 根据 name 属性值获取该值表示节点的对象数组，如 radio 单选框

第 40 道：下列哪一个选项不属于 `document` 对象的方法？（ ）

A. `focus()`

B. `getElementById()`

C. `getElementsByName()`

D. `bgColor()`

答案：D

解析：

`bgColor()` 不属于 `document` 对象的方法；

`focus()` 获得焦点的对象；

`getElementByName()` 方法可返回对拥有指定 ID 的第一个对象的引用。

第 41 道：nodeType 是用来干什么的？空白节点的 nodeType 等于多少？

解析：

nodeType 是用来判断节点类型的，nodeType 等于 3。

第 42 道：找出 ID 为“newsList”的 HTML 元素下的第一个节点，并将其移动到“newsList”的最后。

```
<div id="newsList">
    <p class="a">1</p>
    <p class="b">2</p>
    <p class="c">3</p>
</div>
```

解析：

```
<script type="text/javascript">
var ul = document.getElementById("newsList");
ul.appendChild(ul.firstChild);
</script>
```

第 43 道：怎样添加、移除、复制、创建节点和查找节点？

解析：

appendChild 添加节点；

removeChild 移除节点；

cloneNode 复制节点；

createElement 创建元素；

getElementById、getElementsByName、getElementsByTagName 查找节点。

第 44 道：写出 7 个操作 HTML DOM 对象的方法？

解析：

getElementById(id) 获取带有指定 id 的节点（元素）；

removeChild(node) 删除子节点（元素）；

appendChild(node) 插入新的子节点（元素）；

createElement() 用指定的标记名创建新的 Element 节点对象；

createTextNode() 用指定的文本创建新的文本节点对象；

createAttribute() 用指定名字创建新的 Attr 节点对象。

第 45 道：简述列举文档对象模型 DOM 里 document 的常用查找访问节点的方法？

解析：

document.getElementById 根据元素 id 查找元素；

document.getElementsByTagName 根据元素 name 查找元素；

document.getElementsByTagName 根据标签名查找元素。

第 46 道：下面哪一个是用来追加到指定元素的末尾的（ ）

A. insertAfter() B. append() C. appendTo() D. after()

答案：C

解析：

insertAfter() 是在元素后插入，append() 是向尾部追加。

第 47 道：请说明 JavaScript 中的 nodeName、nodeValue 和.nodeType 的区别。

解析：

1. nodeName

nodeName 属性含有某个节点的名称。

- 元素节点的 nodeName 是标签名称；
- 属性节点的 nodeName 是属性名称；
- 文本节点的 nodeName 永远是 #text；
- 文档节点的 nodeName 永远是 #document。

注意：nodeName 所包含的 XML 元素的标签名称永远是大写的。

2. nodeValue

对于文本节点，nodeValue 属性包含文本。

对于属性节点，nodeValue 属性包含属性值。

nodeValue 属性对于文档节点和元素节点是不可用的。

3..nodeType

nodeType 属性可返回节点的类型。

最重要的节点类型是：

元素类型	节点类型
元素 element	1
属性 attr	2
文本 text	3
注释 comments	8
文档 document	9

第 48 道：实现输出 document 对象中的所有成员的名称和类型。

解析：

```
<script language="javascript">
for(key in document){
    document.write(key + '=== ' + document[key] + '<br />');
}
```

```
</script>
```

1.7 “神马”题都是浮云（className）

第 49 道：请运用 JavaScript 找出所有 className 包含 text 的标签 ，并将它们的背景颜色设置为黄色。

解析：

```
}  
var list=document.getElementsByTagName("a");  
for (i=0;i<list.length;i++){  
    if(list[i].getAttribute("class")== "test"){  
        // 把背景色设置成黄色  
    }  
}
```

1.8 看完这题，花儿都笑了（保留字和关键字）

第 50 道：以下哪个字不属于 JavaScript 的保留字（ ）（多选）

A. with B. parent C. class D. void E. arguments

答案：ABDE

解析：

保留字在某种意义上是为将来的关键字而保留的单词，不能被用来作为变量名和函数名。parent 不属于 JavaScript 的保留字。

第 51 道：以下哪个字不属于 JavaScript 的关键字（ ）（多选）

A. abstract B. import C. return D. void E. if

答案：AB

解析：

JavaScript 的关键字对 JavaScript 的编译器有特殊的意义，它们用来表示一种数据类型，或者表示程序的结构等，关键字不能作为变量名、方法名、类名、包名。

第 52 道：关键字与保留字的区别？

解析：

从字面含义上理解，保留字是语言中已经定义过的字，使用者不能再将这些字作为变量名或过程名使用。而关键字则指在语言中有特定的含义，成为语

法中一部分的那些字。在一些语言中，一些保留字可能并没有应用于当前的语法中，这就成了保留字与关键字的区别。一般出现这种情况可能是由于考虑了扩展性。例如，JavaScript 有一些未来保留字，如 abstract、double、goto 等。

1.9 我懂的题，你懂吗（循环语句）

第 53 道：JavaScript 有几种循环语句的写法？

解析：

- (1) for 循环结构语句（可以嵌套）。
- (2) for-in 循环结构语句。
- (3) while 循环语句结构。
- (4) do-while 循环语句结构。

第 54 道：下列 JavaScript 代码执行后，iNum 的值是 _____

```
var iNum=0;
for(var i=1;i<10;i++){
  if(i%5==0){
    Continue;
  }
  iNum++;
}
```

答案：8

解析：

continue 只是退出当前循环，进入下一次循环。要详细了解 continue 和 break 的用法。

第 55 道：下列 JavaScript 代码执行后，alert() 的结果是什么？

```
for(i=0;j=0;i<10,j<6;i++;j++){
  k=i+j;
}
alert(k);    // 返回什么
```

解析：

alert() 的结果是 18，这个循环要进行 10 次，第一次，j 和 i 均为 0……第十次，j=9，i=9，自加 1，进行条件判断，10<10 不成立，循环停止。

第2章



能文能武，身怀绝技 [JavaScript中级面试题]

恭喜你已修炼到中级了，相信已经有一定的能力可以独自处理一些事情，然而你还需要修炼，在这里表达式带你出神入化，string 对象引领你步入巅峰，数组让你大彻大悟，函数让你得到飞升。

2.1 小考题，大味道（表达式）

第 56 道：请选择结果为真的表达式（ ）

- A. `null instanceof Object`
- B. `null === undefined`
- C. `null == undefined`
- D. `NaN == NaN`

答案：C

解析：

`null` 确实可以理解为原始类型，但不能当 `Object` 理解。`null`、`int`、`float` 等这些用关键字表示的类型，都不属于 `Object`。至于可以把 `null` 作为参数，只是特殊规定而已。对象的引用代表的是一个内存的值，`null` 是一个空引用，可以理解为内存的值为 0。

第 57 道：请说明 JavaScript 中 “==” 和 “===” 的区别。

解析：

在 JavaScript 中，“==” 直接比较两个变量的值，但 “===” 则比较两个变量的值和类型，前者在对不同类型比较时 JavaScript 会做出相应的类型转换，转换之后若相等返回 `true`，否则返回 `false`。

第 58 道：请选择结果为真的表达式（ ）（多选）

- A. `null instanceof object`
- B. `null === undefined`
- C. `null == undefined`
- D. `NaN == NaN`
- E. `null == null`
- F. `null === null`

答案：CEF

解析：

除“空”之外所有的字符串，以及除 0 之外所有的数字，转换为布尔型都是 `true`。

第 59 道：简要描述下列符号在 JavaScript 中的作用，“||”，“eval”，“call”。

解析：

“||”是或的意思。

“eval”是把一个字符串当作一个 JavaScript 表达式一样去执行。

“call”可以用来代替另一个对象调用一个方法。call 方法可将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。

第 60 道：JavaScript 中表达式 `parseInt("X8X8")+parseFloat("8")` 的结果是（ ）

A. 8+8 B. 88 C. 16 D. "8"+8

答案：C

解析：

`parseInt()` 提取整数部分，`parseFloat()` 提取浮点数部分。

第 61 道：统计从 1 至 400 亿之间的自然数中含有多少个 1？比如从 1 ~ 21，1、10、11、21 这 4 个自然数中共含有 5 个 1。

解析：

(1) 定义初始个数。

(2) 把每个数拆分成单个数字，如 123 拆分成 "1","2","3"。

(3) 用 `toString()` 方法进行判断，如果包含 1，则统计 1 的个数。

第 62 道：分析下面的 JavaScript 代码：

```
x=11;        y='number';        m=x+y;
```

m 的值为多少？

答案：11number

解析：

在 `parseFloat` 中，相加的话，加号表示连接。

第 63 道：什么是三元条件语句？

解析：

三元条件语句是“条件？结果 1：结果 2；”，这里把条件写在问号(?)的前面，后面跟着用冒号(:)分隔结果 1 和结果 2。满足条件时为结果 1，否则为结果 2。

2.2 让人叹为观止的考题（String 对象）

第 64 道：在 JavaScript 中，String 对象的方法不包括（ ）

A. `charAt()` B. `substring()` C. `toUpperCase()` D. `length`

答案：D

解析：

charAt() 从某个字符串取得具体的字符；

substring() 截取字符串；

toUpperCase() 将字符串中的小写字母转换成大写字母；

length 不是方法。

第 65 道：请编写代码扩展 JavaScript 的 string 对象，让其拥有一个新的方法 killpoint() 来删除字符串中的所有英文句号 “.”，请用尽量少的代码实现。

解析：

```
String.prototype.killpoint=function(){
    return this.replace(/\./g, '');
}
```

第 66 道：对 string 对象进行扩展，使其具有删除前后空格的方法。

解析：

```
String.prototype.deleteSpace = function(){
    var str = this;    // 提取需要操作的字符串
    while(str[0] == " "){ // 删除前面的空格
        str = str.substring(1);
    }
    while(str[str.length - 1] == " "){ // 删除后面的空格
        str = str.substring(0, str.length-1);
    }
    return str;
}
```

第 67 道：如何扩展 JavaScript 中原生的 String 对象？String 对象中的哪些方法可返回字符串的指定部分？

解析：

(1) 用 prototype 添加方法。

(2) substring() 函数是返回截取之后的字符串，不会对原字符串进行修改。

```
// 获取字符数组
String.prototype.toCharArray = function() {
    return this.split("");
}
// 获取 N 个相同的字符串
String.prototype.repeat = function(num) {
    var tmpArr = [];
    for ( var i = 0; i < num; i++)
        tmpArr.push(this);
    return tmpArr.join("");
}
// 逆序
```

```
String.prototype.reverse = function() {
    return this.split("").reverse().join("");
}
// 测试是否是数字
String.prototype.isNumeric = function() {
    var tmpFloat = parseFloat(this);
    if (isNaN(tmpFloat))
        return false;
    var tmpLen = this.length - tmpFloat.toString().length;
    return tmpFloat + "0".Repeat(tmpLen) == this;
}
// 测试是否是整数
String.prototype.isInt = function() {
    if (this == "NaN")
        return false;
    return this == parseInt(this).toString();
}
// 合并多个空白为一个空白
String.prototype.resetBlank = function() {
    return this.replace(/s+/g, " ");
}
// 除去左边空白
String.prototype.LTrim = function() {
    return this.replace(/^s+/g, "");
}
// 除去右边空白
String.prototype.RTrim = function() {
    return this.replace(/s+$/g, "");
}
// 除去两边空白
String.prototype.Trim = function() {
    return this.replace(/(^s+)|(s+$/g, "");
}
// 保留数字
String.prototype.getNum = function() {
    return this.replace(/[^\d]/g, "");
}
// 保留字母
String.prototype.getEn = function() {
    return this.replace(/[^\A-Za-z]/g, "");
}
// 保留中文
```

```

String.prototype.getCn = function() {
    return this.replace(/^[^u4e00-u9fa5uf900-ufa2d]/g, "");
}
// 得到字节长度
String.prototype.getRealLength = function() {
    return this.replace(/^[^x00-xff]/g, "--").length;
}
// 从左截取指定长度的字串
String.prototype.left = function(n) {
    return this.slice(0, n);
}
// 从右截取指定长度的字串
String.prototype.right = function(n) {
    return this.slice(this.length - n);
}
// html 编码
String.prototype.htmlEncode = function() {
    var re = this;
    var q1 = [ /x26/g, /x3C/g, /x3E/g, /x20/g ];
    var q2 = [ "&", "<", ">", " " ];
    for ( var i = 0; i < q1.length; i++)
        re = re.replace(q1[i], q2[i]);
    return re;
}
// Unicode 转化
String.prototype.ascW = function() {
    var strText = "";
    for ( var i = 0; i < this.length; i++)
        strText += "&#" + this.charCodeAt(i) + ";";
    return strText;
}

```

String 对象的方法 slice()、substring() 和 substr() (不建议使用) 都可返回字符串的指定部分。

2.3 让人绞尽脑汁的考题 (Math 对象)

第 68 道：请用 JavaScript 实现获取 5 个 0 ~ 99 之间不相同的随机数。

解析：

思路：首先创建一个 0 ~ 99 的数组，每次取一个数，然后去除数组中取出的这个数，这样就可以实现永不重复。

公式: `document.write(Math.floor(Math.random()*10+1));`

使用函数:

(1) `Math.random()`; 结果为 0 ~ 1 间的一个随机数 (包括 0, 不包括 1)。

(2) `Math.floor(num)`; 参数 `num` 为一个数值, 函数结果为 `num` 的整数部分。

(3) `Math.round(num)`; 参数 `num` 为一个数值, 函数结果为 `num` 四舍五入后的整数。

第 69 道: 请给出下面 a、b、c 的输出结果 _____。

```
var a=parseInt("11",2);
var b=parseInt("02",10);
var c=parseInt("09/08/2009");
```

答案: 3, 2, 9

解析:

`parseInt()` 的用法: `parseInt()` 函数可解析一个字符串, 并返回一个整数。

语法: `parseInt(string, radix);`

`c` 的首字符是 0, 一般想法是按 8 进制进行解析。但是我们发现第二个字符 9 已经不是 8 进制数, 也就是说, 9 是第一个不能解析的字符。`parseInt("09/08/2009")` 也就变成了 `parseInt("0")`; 结果再明显不过了, 是 0。

如果我们稍加变化 `parseInt("0119/08/2009")`, 首字符是 0, 按 8 进制解析, 同样读到 9 时不能再解析了, 也就变成了 `parseInt("011")`, 结果也很明显, 是 9。

第 70 道: 编写 JavaScript 脚本, 生成 0 ~ 7 之间的随机整数。

解析:

公式: `document.write(Math.floor(Math.random()*7));`

使用函数:

(1) `Math.random()`; 结果为 0 ~ 1 间的一个随机数 (包括 0, 不包括 1)。

(2) `Math.floor(num)`; 参数 `num` 为一个数值, 函数结果为 `num` 的整数部分。

(3) `Math.round(num)`; 参数 `num` 为一个数值, 函数结果为 `num` 四舍五入后的整数。

第 71 道: 使用 JavaScript 求两个数的最大公约数。

解析:

```
function f(num1, num2){
    for(var i = Math.min(num1, num2); i > 0; i--){
        if(num1%i == 0 && num2%i == 0)
            return i;
    }
}
```

第 72 道: 获取一个 1 ~ 50 的随机不重复数组。

解析:

```
var arr=[];
```

```

var number=50;
for(var i=1;i<=number;i++){
    arr.push(i);
}
var result=[];
for(var j=number-1;j>=0;j--){
    var rand=Math.ceil(Math.random()*j);
    result.push(arr.splice(rand,1));
}

```

第 73 道：Math.ceil()、Math.floor() 和 Math.round() 三个方法都是四舍五入，请问有什么区别？Math.round(-11.5) 的值是多少？

解析：

- (1) Math.ceil() 用于向上取整。
- (2) Math.floor() 用于向下取整。
- (3) Math.round() 用于四舍五入取整。
- (4) Math.round(-11.5) 的值是 -11。

2.4 令人陶醉的考题（数组）

第 74 道：在 JavaScript 里，下列选项中不属于数组方法的是（ ）

- A. sort() B. length() C. concat() D. reverse()

答案：B

解析：

sort() 方法用于对数组的元素进行排序。

length() 返回的是字符串的长度。

concat() 方法用于连接两个或多个数组。

reverse() 方法用于颠倒数组中元素的顺序。

第 75 道：请见如下代码：

```

var emp=new Array(3);
for(var i in emp)

```

以下答案中能与 for 循环代码互换的是（ ）

- A. for(var i=0;i<emp;i++)
- B. for(var i=0;i<Array(3);i++)
- C. for(var i=0;i<emp.length();i++)
- D. for(var i=0;i<emp.length;i++)

答案：D

解析：

for/in 循环遍历对象的属性，而 length 不是方法，也不是对象的属性。

第 76 道：下列声明数组的语句中，错误的选项是（ ）

- A. `var arry=new Array()` B. `var arry=new Array(3)`
C. `var arry[]=new Array(3)(4)` D. `var arry=new Array("3","4")`

答案：C

解析：

创建一个数组：

```
arrayObj = new Array(); // 创建一个数组
arrayObj = new Array([size]); // 创建一个数组并指定长度，注意不是
上限，是长度
arrayObj = new Array([element0[, element1[, ...[,
elementN]]]]); // 创建一个数组并赋值
arrayObj = [element0, element1, ..., elementN]; // 创建一个数
组并赋值的简写，注意这里中括号不表示可省略
```

第 77 道：请编写尽可能简洁的 JavaScript 代码，找到在第一个数组 `array1` 中出现，而在第二个数组 `array2` 中没有出现的数字。

解析：

`indexOf()` 判断数字是否出现，`join()` 用于把数组中的所有元素放入一个字符串。元素是通过指定的分隔符进行分隔的。

```
function findNullofNum(arr1,arr2) {
    var str = arr2.join("");
    var result = [];
    for (var i = 0,x =0;i<arr1.length;i++) {
        if (str.indexOf(arr1[i]) == -1)
            result[x]=arr1[i];
        x++;
    }
    return result;
}
var b = findNullofNum(arr1,arr2);
```

第 78 道：编写函数，用于过滤一个数组内重复的元素，并用这些元素重构一个新数组，新数组内也不能有重复元素。

```
var arrNum=[1,4,1,1,3,3,4,6,7,8,3,7,0,11,22,22];
```

解析：

将数组中重复的元素去掉，并将各元素放到新的数组中。原理：将重复的元素替换为空。

```
var str = arrNum.join(',');
var newArr = [];
for(var i = 0; i < arrNum.length; i++){
    if(str.indexOf(arrNum[i]) != -1){
```

```

        newArr[newArr.length] = arr[i];
    }
    while(str.indexOf(arrNum[i]) != -1){
        str = str.replace(arrNum[i], ' ');
    }
}

```

第 79 道：现有一个数组（元素为数字，并且有可能重复），请给 Array.prototype 增加一个方法（方法名自取），该方法能去掉数组中全部最大和最小的数字。

解析：

数组最大值

```

Array.prototype.max=function
{
    return Math.max.apply(null,this);
}

```

数组最小值

```

Array.prototype.min=function
{
    return Math.min.apply(mull,this);
}

```

第 80 道：给定一个数组实现字符串反转，要求原地实现。

解析：

使用循环数组的方法：

```

var arr = new Array();
function myreverse(arr) {
    for (var i = 0; i < arr.length / 2; i++) {
        var temp = arr[i]; // 交换变量
        arr[i] = arr[arr.length - i - 1];
        arr[arr.length-i-1]=temp;
    }
}

```

第 81 道：如何准确判断一个 JavaScript 对象是数组？

解析：

标准方法：

```

if(object.prototype.toString.call(a) == '[object Array]'){
    // 是数组对象
}

```

常用方法：

```

if(a.constructor == Array){
    // 是数组对象；
}

```

第 82 道：在如下数组的第二个元素后插入一个元素 3。

```
var arr=[1,2,4,5,6];
```

解析：

语法：arr.splice(index,howmany,element1,...,elementN);

参数解释：

- (1) index——从该下标开始删除。
- (2) howmany——删除指定数量的元素。
- (3) elements——插入的元素。

作用：从指定位置删除部分元素并增加新的元素

- (1) 该方法返回值是被删除的元素组成的数组。
- (2) splice 是直接对数组进行操作，而 slice 函数则是取 arr 的一段元素，原数组不变。

```
arr.splice(2,0,3)
```

第 83 道：以空格字符串作为分隔字符串，将如下字符串拆分成数组（每个元素不能含有空格字符）。

```
var str='ab c 20 d e f g 123';
```

解析：

```
str.split(/\s+/);
```

语法：stringObject.split(separator,howmany);

参数：separator

描述：必需。字符串或正则表达式，从该参数指定的地方分割 stringObject。

参数：howmany

描述：可选。该参数可指定返回的数组的最大长度。如果设置了该参数，返回的子串不会多于这个参数指定的数组。如果没有设置该参数，整个字符串都会被分割，不考虑它的长度。

第 84 道：怎么判断一个对象为数组类型？

解析：

typeof 操作符：

typeof 可以解决大部分的数据类型判断，它是一个一元运算，放在一个运算值之前，其返回值为一个字符串。该字符串说明运算数的类型，所以判断某个是否为 String 类型，可以直接 if(typeof(你的值)=="string"){}

instanceof 操作符：

instance，顾名思义，实例、例子。所以 instanceof 用于判断一个变量是否为某个对象的实例，是一个三目运算式，这是它和 typeof 最实质上的区别。

```
a instanceof b?alert("true"):alert("false") // 注意b值是你想要判断的那种数据类型，不是一个字符串，比如 Array 对象的 constructor 属性
```


`constructor` 属性返回对创建此对象的数组函数的引用就是返回对象相对应的构造函数。从定义上来说，已跟 `instanceof` 不太一致，但效果都是一样的。

如：

```
(a instanceof Array)    //a 是否为 Array 的实例? true or false
(a.constructor == Array) // a 实例所对应的构造函数是否为 Array?
true or false
```

第 85 道：`var aPush = [1, 2, 3];`

```
console.log("len: "+aPush.push(4)+",push 后数组: "+aPush);
```

解析：

`len: 4; push 后数组: 1, 2, 3, 4。`

`push()` 方法可向数组的末尾添加一个或多个元素，并返回新的长度。

语法：`arrayObject.push(newelement1,newelement2,...,newelementX)`

参数：`newelement1`

描述：必需。要添加到数组的第一个元素。

参数：`newelement2`

描述：可选。要添加到数组的第二个元素。

参数：`newelementX`

描述：可选。可添加多个元素。

返回值：把指定的值添加到数组后的新长度。

第 86 道：`var aPop = ["a", "b", "c"];`

```
console.log(" 返回: "+aPop.pop()+", pop 后数组: "+aPop);
```

解析：

返回：`c; pop 后数组: a, b。`

`pop()` 方法用于删除并返回数组的最后一个元素。

语法：`arrayObject.pop()`

返回值：`arrayObject` 的最后一个元素。

第 87 道：`var aUnshift = [1, 2, 3];`

```
console.log("len: "+aUnshift.unshift(0)+", unshift 后数组:
"+aUnshift);
```

解析：

`len: 4; unshift 后数组: 0, 1, 2, 3。`

`unshift()` 方法可向数组的开头添加一个或更多元素，并返回新的长度。

语法：`arrayObject.unshift(newelement1,newelement2,...,newelementX)`

参数：`newelement1`

描述：必需。向数组添加的第一个元素。

参数：`newelement2`

描述：可选。向数组添加的第二个元素。

参数: newelementX

描述: 可选。可添加若干个元素。

第 88 道: `var aShift = ["a", "b", "c"];`

```
console.log(" 返回: "+aShift.shift()+" , shift 后数组: "+aShift);
```

解析:

返回: a, shift 后数组: b, c。

shift() 方法用于把数组的第一个元素从中删除, 并返回第一个元素的值。

语法: `arrayObject.shift()`

返回值: 数组原来的第一个元素的值。

说明: 如果数组是空的, 那么 shift() 方法将不进行任何操作, 返回 undefined 值。请注意, 该方法不创建新数组, 而是直接修改原有的 arrayObject。

第 89 道: `var sSplice = [1, 2, 3], sSplice2 = [1, 2, 3], sSplice3 = [1, 2, 3];`

```
sSplice.splice(0, 0, 0);
console.log(sSplice); // 添加
sSplice2.splice(0, 1);
console.log(sSplice2); // 删除
sSplice3.splice(0, 1, "a");
console.log(sSplice3); // 替换
```

解析:

`[0, 1, 2, 3]`

`[2, 3]`

`["a", 2, 3]`

Splice() 方法从数组中添加 / 删除项目, 然后返回被删除的项目。

注释: 该方法会改变原始数组。

语法: `arrayObject.splice(index, howmany, item1, ..., itemX)`

参数: index

描述: 必需。整数, 规定添加 / 删除项目的位置, 使用负数可从数组结尾处规定位置。

参数: howmany

描述: 必需。要删除的项目数量。如果设置为 0, 则不会删除项目。

参数: item1, ..., itemX

描述: 可选。向数组添加的新项目。

第 90 道: `console.log([1, 2, 3].slice(0, 2));`

解析:

`[1, 2]`

slice() 方法可从已有的数组中返回选定的元素。

语法: `arrayObject.slice(start, end)`

参数: start

描述: 必需。规定从何处开始选取。如果是负数，那么它规定从数组尾部开始选取。也就是说，-1 指最后一个元素，-2 指倒数第二个元素，以此类推。

参数: end

描述: 可选。规定从何处结束选取。该参数是数组片段结束处的数组下标。如果没有指定该参数，那么切分的数组包含从开始到结束的所有元素。如果这个参数是负数，那么它规定的是从数组尾部开始算起的元素。

第 91 道: `console.log(["a", "c", "d", "b"].sort());`
`console.log(["10", "1", "2", "5"].sort(function(v1, v2){`
`return v1 - v2;`
`}));`

解析:

`["a", "b", "c", "d"]`

`["1", "2", "5", "10"]`

`push()` 方法可向数组的末尾添加一个或多个元素，并返回新的长度。

语法: `arrayObject.push(newelement1,newelement2,...,newelementX)`

参数: newelement1

描述: 必需。要添加到数组的第一个元素。

参数: newelement2

描述: 可选。要添加到数组的第二个元素。

参数: newelementX

描述: 可选。可添加多个元素。

第 92 道: 将数组 `["a","b"]` 和 `["c","d"]` 合并，并且删除第二个元素。

解析:

合并数组:

`Array.prototype.push.apply (mergeTo,mergeFrom);`

删除数组中某个值:

`splice (index,len,[item]);`

注释: 该方法会改变原始数组。

`splice` 有 3 个参数，它可以用来替换 / 删除 / 添加数组内某一个或者几个值。

- index: 数组开始下标。
- len: 替换 / 删除的长度。
- item: 替换的值，删除操作的话，item 为空。

第 93 道: Javascript 数组方法 `sort()` 的作用是 _____

解析:

`sort()` 方法用于实现对数组元素的排序，该方法默认按照数组项 ASCII 码字

符的顺序升序排列。在执行过程中并不会创建新的 Array 对象。

如果为 sort() 参数提供了一个函数，那么该函数必须返回下列值之一：

- 负值（如果所传递的第一个参数比第二个参数小）；
- 零（如果两个参数相等）；
- 正值（如果第一个参数比第二个参数大）。

第 94 道：写出一个 JavaScript 的函数，实现对一个数组去重的功能。

解析：

思路：

- （1）构建一个新的数组存放结果。
- （2）for 循环中每次从原数组中取出一个元素，用这个元素循环与结果数组对比。

- （3）若结果数组中没有该元素，则存到结果数组中

```
Array.prototype.unique1 = function(){
    var res = [this[0]];
    for(var i = 1; i < this.length; i++){
        var repeat = false;
        for(var j = 0; j < res.length; j++){
            if(this[i] == res[j]){
                repeat = true;
                break;
            }
        }
        if(!repeat){
            res.push(this[i]);
        }
    }
    return res;
}
```

第 95 道：请写出如下 JavaScript 代码的运行结果。

```
var name="zhangsan";
function GetName(){
    var arr=[1,2,3];
    var name="list"+arr;
    document.write(name+"</br>");
}
GetName();
document.write(name);
```

答案：list1,2,3 zhangsan

解析：

用于显式指定函数的调用对象方法有三个：

(1) 如果一个函数被赋为一个对象的属性值，这个函数只能通过该对象来访问（但并非是说该函数只能被该对象调用），通过该对象调用这个函数的方式类似以面向对象编程语言中的方法调用（实际上在 JavaScript 中也习惯使用方法这种称呼）。

(2) 任意指定函数的调用对象，在某个语法环境中，如果可以同时访问到函数 fun 和对象 obj，只要你愿意，可以指定通过 obj 对象来调用 fun 函数。指定方法有两种：call 方法和 apply 方法。

(3) 另一种比较容易疏忽的错误是，在 A 对象的方法中，执行了使用 B 对象的方法调用，试图在 B 对象的方法里使用 this 来访问 A 对象。这在各种回调函数中比较常见，最常见的情形就是 Ajax 回调函数中使用 this。

第 96 道：请写出如下 JavaScript 代码片段的运行结果。

```
var my_arr=[];
for(var i=0;i<=5;i++){
    my_arr.push(i*(i+1));
}
var val=0;
while(val=my_arr.pop()){
    document.write(val+" ");
}
```

答案：数组为：2,6,12,20,30

解析：

my_arr.pop() 一个一个删除数组最后一个元素，当删除到第一个 0 之后，返回的值为 undefined，那么就跟下面的 while 条件不符合了。故最后一个 0 是不会输出的。

while 语句总是先检测循环表达式，所以它的循环体可能一次都不执行。

while 循环会在指定条件为真时，循环执行代码块。

第 97 道：JavaScript 中给全部都是数字元素的数组排序的原生方法是 _____，其中使用的是 _____ 的排序方法？

答案：sort()、自定义排序规则。

解析：

JavaScript 中将数组排序的方法有很多，其中 sort() 方法默认是按字符来排序的，所以在对数字型数组排序时，不可想当然地以为会按数字大小排序。要想改变默认的 sort 行为（即按字符排序），可以自行指定排序规则函数。

下面写一段脚本来验证一下：

```
<script type="text/JavaScript">
    $(document).ready(function() {
        var list = [1,2,12,43,5,64,7];
```

```

        $('p').html(list.sort(function(a,b){
            return a-b;
        }).join("<br/>"));
    });
</script>

```

执行脚本后页面中会换行依次显示 1、2、5、7、12、43、64。

第 98 道：join() 和 split() 的区别？

解析：

join 函数获取一批字符串，然后用分隔符字符串将它们连接起来，从而返回一个字符串。split 函数获取一个字符串，然后在分隔符处将其断开，从而返回一批字符串。但是，这两个函数之间的主要区别在于，join 可以使用任何分隔符字符串将多个字符串连接起来，而 split 只能使用一个字符分隔符将字符串断开。

简单地说，split 是把一串字符（根据某个分隔符）分成若干个元素存放在一个数组里，而 join 是把数组中的字符串连成一个长串，可以大体上认为是 split 的逆操作。

第 99 道：console.log([1,2,3].length);

答案：3

解析：

length 属性表示数组的长度，即其中元素的个数。因为数组的索引总是从 0 开始的，所以一个数组的上下限分别是 0 和 length-1。

第 100 道：console.log([1, 2, 3].concat("a", "b")) 的结果是什么？

答案：[1, 2, 3, "a", "b"]

解析：

concat() 方法用于连接两个或多个数组。

该方法不会改变现有的数组，而仅仅会返回被连接数组的一个副本，返回一个新的数组。该数组是通过把所有 arrayX 参数添加到 arrayObject 中生成的。如果要进行 concat() 操作的参数是数组，那么添加的是数组中的元素，而不是数组。

第 101 道：console.log([2013,10,1].join("")) 的结果是什么？

答案：2013101

解析：

join() 方法用于把数组中的所有元素放入一个字符串。

元素是通过指定的分隔符进行分隔的。

返回一个字符串，此字符串由包含在数组中的许多子字符串连接创建。

语法：join(list[, delimiter])

join 函数的语法有以下参数：

list，必选。包含要连接的子字符串一维数组。

delimiter, 可选。在返回字符串中用于分隔子字符串的字符。如果省略, 将使用空字符 (""). 如果 delimiter 是零长度字符串, 则在同一列表中列出全部项, 没有分界符。

第 102 道: slice() 与 splice() 分别具备什么功能? 有什么区别?

解析:

slice() 方法: 接收 1 个或 2 个参数, 即要提取的项的起始位置和结束位置。如果只有一个参数, 该方法返回从该位置开始到数组结尾的所有项, 如果有 2 个参数, 该方法返回第一个位置和第二个位置的所有项, 不包括第二个位置的项。

splice() 方法: 把数据项插入数组中部。调用该方法时传入的参数不同, 就用不同方法删除。传入 2 个参数, 这 2 个参数是要删除的第一个项的位置和要删除的项的个数, 替换而不删除; 传入 3 个参数, 这 3 个参数是起始位置和 0 (要删除的个数), 要插入的项, 替换并删除。

第 103 道: 请写出一个函数, 功能是删除数组的指定下标元素。

解析:

```
array.pop(); // 删除最后一个元素, 并返回该元素
array.shift(); // 删除第一个元素, 数组元素位置自动前移, 返回被删除的元素
array.splice(start, delCount); // 从 start 的位置开始向后删除 delCount 个元素

Array.prototype.del=function(index){
    if(isNaN(index)||index>=this.length){
        return false;
    }
    for(var i=0,n=0;i<this.length;i++){
        if(this[i]!==this[index]){
            this[n++]=this[i];
        }
    }
    this.length-=1;
};
```

第 104 道: 请写一个函数 removeVoid(arr), 删除该数组中值为 “null, undefined” 的项, 返回原数组。

解析:

```
function ClearNullArr(arr)
{
    for(var i=0,len=arr.length;i<len;i++){
        if(!arr[i]||arr[i]==''||arr[i] === undefined)
        {
            arr.splice(i,1);
            len--;
        }
    }
}
```

```
        i--;
    }
}
return arr;
}
```

第 105 道：数组方法 pop()、push()、unshift() 和 shift() 分别具备什么功能？

解析：

pop(): 从集合中把最后一个元素删除，并返回这个元素的值。

push(): 在集合中添加元素，并返回新的长度。

unshift(): 在集合开头添加一个或更多元素，并返回新的长度。

shift(): 从集合中把第一个元素删除，并返回这个元素的值。

第 106 道：请给 Array 本地对象增加一个原型方法，它的用途是删除数组中重复的条目，并将数组返回。

解析：

```
<script type="text/javascript">
    Array.prototype.delrepeat = function(){
        var arr = this;
        var _arr = new Array();
        for (var i in arr){
            if (i == 'delrepeat')continue;
            if (_arr.length == 0) _arr.push(arr[i]);
            for (var j = 0; j < _arr.length; j++){
                if (arr[i] == _arr[j]){
                    break;
                }
                if(j > _arr.length - 2) _arr.push(arr[i]);
            }
        }
        return _arr;
    }
</script>
```

2.5 令人难过的考题（函数）

第 107 道：在 JavaScript 中如何写定时调用函数 foo()？

解析：

```
function foo(){ alert("qdjhu");
    setTimeout(foo,1000);
```

第 108 道：在 JavaScript 中如何规避多人开发函数重名问题？

解析：

- (1) 命名空间，根据不同开发人员开发的功能在函数前加前缀；
- (2) 立即执行函数模式（即时对象初始化），避免污染全局环境。

第 109 道：请分别描述 JavaScript 中 prototype、constructor、this、argument 的含义。

解析：

prototype：prototype 的行为类似于 C++ 中的静态域，将一个属性添加为 prototype 的属性，这个属性将被该类型创建的所有实例所共享，但是这种共享是只读的。在任何一个实例中只能够用自己的同名属性覆盖这个属性，而不能够改变它。换句话说，对象在读取某个属性时，总是先检查自身域的属性表，如果有这个属性，则会返回这个属性，否则就去读取 prototype 域，返回 prototype 域上的属性。另外，JavaScript 允许 prototype 域引用任何类型的对象。因此，如果对 prototype 域的读取依然没有找到这个属性，则 JavaScript 将递归地查找 prototype 域所指向对象的 prototype 域，直到这个对象的 prototype 域为它本身或者出现循环为止。

constructor：即构造函数，在对象创建或者实例化时被调用的方法。通常使用该方法来初始化数据成员和所需资源。构造器 constructor 不能被继承，因此不能重写 overriding，但可以被重载 overloading。对象的 constructor 属性返回创建该对象的函数的引用。

this：在 JavaScript 中，this 通常指向的是正在执行的函数本身，或者是指向该函数所属的对象（运行时）。当我们在页面中定义函数 doSomething() 时，它的 owner 是页面，或者是 JavaScript 中的 window 对象（或 global 对象）。对于一个 onclick 属性，则为它所属的 HTML 元素所拥有，this 应该指向该 HTML 元素。

argument：所有的函数都有属于自己的一个 arguments 对象，它包括了函数所要调用的参数。它不是一个数组，如果用 typeof arguments，那么返回的是 object。虽然我们可以用调用数据的方法来调用 arguments。比如 length、index 方法。但是数组的 push 和 pop 对象是不适用的。

第 110 道：写一个函数，参数为一个元素，返回指定元素的第一个子元素，要求兼容 IE6/IE7/IE8/Firefox/Safari/Chrome，函数越简单越好，代码大概如下：

```
function getfirst(el) {
    //...
}
```

解析：

```
function getfirst(el) {
    var nodes=el.children;    // 获取元素下所有的子节点
```

```
return nodes.length!=0?nodes[0]:null;    // 判断子节点是否存在, 如果存在, 那么返回该元素下第一个子节点, 不存在则返回空
}
```

第 111 道：写出下面函数 console 的值，并说出为什么？

```
var b=1;
function c(){
    console.log(b);
    if(!b){
        var b=2;
    }
    console.log(b);
}
c();
```

解析：

此题考查的是 JS 局部变量声明前置。也就是说，函数中语句 `var b=2; b` 声明在函数的最前面，但赋值语句不变，因此答案为先输出 `undefined` 再输出 `2`。

第 112 道：看下面一段代码，请写出结果。

```
function b(){
    a=10;
    alert(arguments[1]);
}
b(1,2,3);
```

答案：弹出的结果是 2。

解析：

`arguments` 对象不能显式创建，`arguments` 对象只有函数开始时才可用。函数的 `arguments` 对象并不是一个数组，访问单个参数的方式与访问数组元素的方式相同。索引 `n` 实际上是 `arguments` 对象的 `0 ~ n` 属性的其中一个参数。

`arguments` 对象和 `function` 对象紧密地联系在一起。也就是说，每一个函数都有自己的 `arguments` 属性。

通过 `arguments` 属性（相对于 `function` 来说），函数可以处理可变数量的参数。`arguments` 对象的 `length` 属性包含了传递给函数的参数的数目。对于 `arguments` 对象所包含的单个参数，其访问方法与数组中所包含的参数的访问方法相同，用 `0` 到 `arguments.length-1` 来枚举每一个元素。`callee` 属性是 `arguments` 对象的一个成员，仅当相关函数正在执行时才可用。`callee` 属性的初始值就是正被执行的 `function` 对象，这允许匿名的递归函数。

第 113 道：简述 JavaScript 封装。

解析：

封装可以被定义为对对象的内部数据表现形式和实现细节进行隐藏。通过

封装可以强制实施信息隐藏。

在 JavaScript 中，并没有显式的声明私有成员的关键字等。所以要想实现封装或信息隐藏，就需要从另外的思路出发。我们可以使用闭包的概念来创建只允许从对象内部访问的方法和属性，来达到封装的要求。

第 114 道：下面 JavaScript 代码的运算结果是 2 还是 undefined？请阐述原因。

```
function show() {
    var b=1;
    a=++b;
}
show();
alert(a);
```

答案：2。

解析：

++b 先返回递增以后的值。

第 115 道：判断以下两段代码的正误，并阐述错误代码为何报错。

sum(1);	sum(1);
function sum() {	var sum=function() {
alert(sum);	alert(sum);
}	}

解析：

JavaScript 中需要创建函数的话，有两种方法：函数声明、函数表达式。各自写法如下。

方法一：函数声明。

```
function foo() {}
```

方法二：函数表达式。

```
var foo = function () {};
```

另外，还有一种自执行函数表达式，主要用于创建一个新的作用域，在此作用域内声明的变量不会和其他作用域内的变量冲突或混淆，大多是以匿名函数的方式存在，且立即自动执行：

```
(function () {
    // var x = ...
})();
```

此种自动执行函数表达式归类于以上两种方法中的第二种，也算是函数表达式。

第 116 道：JavaScript 中，有一个函数，在执行对象查找时，永远不会去查找原型，这个函数是什么？

解析：

JavaScript 中 hasOwnProperty 函数方法是返回一个布尔值，并指出一个对象

是否具有指定名称的属性。使用方法: `object.hasOwnProperty (proName)`。其中, 参数 `object` 是必选项, 一个对象的示例, `proName` 是必选项, 一个属性名称的字符串值。如果 `object` 具有指定名称的属性, 那么 JavaScript 中 `hasOwnProperty` 函数方法返回 `true`; 反之则返回 `false`。此方法无法检查该对象的原型链中是否具有该属性, 该属性必须是对象本身的一个成员。

第 117 道: 请写一个函数 `closest(element,className)`, 传入 DOM 对象及 CSS 名称, 或者标签名称, 查找到离它自身最近的父节点。

解析:

```
<script>
    function closest(element,classname){
        if(element){    // 判断元素是否存在
            if(element.className==classname){    // 如果存在,
                则判断当前元素的 CSS 名称是否等于接收的 CSS 名称
                return element.parentNode;    // 如果等于, 返回
                当前元素最近的父节点
            }else{
                return element.parentNode=null;
                // 否则返回 "null"
            }
        }else{
            console.log(" 该元素不存在 ");    // 否则该元素不存在
        }
    }
</script>
```

第 118 道: 请写一个函数 `getParameters()` 来获取浏览器地址栏 URL 全部参数, 并返回一个 JSON 串。

解析:

采用正则表达式获取地址栏参数:

```
function getParameters(name){
    var reg = new RegExp("(^|&)" + name + "=(^[&]*)(&|$)");
    var r = window.location.search.substr(1).match(reg);
    if(r!=null)return  unescape(r[2]); return null;
}
// 调用方法
alert(GetQueryString(" 参数名 1"));
alert(GetQueryString(" 参数名 2"));
alert(GetQueryString(" 参数名 3"));
```

第 119 道: 请写一个函数来验证电子邮件的格式是否正确。

解析:

```
function CheckMail($mailaddress){
```

```

    if(ereg('^[a-zA-Z0-9_-.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+\$',
$mailaddress)){
        return true;
    }else{
        return false;
    }
}

```

对于正则的解释：

^[a-zA-Z0-9_-.] 是至少由一个字符、数字、下画线、连字符、点为开始的字符串；\$ 更多。

第 120 道：写一个获取非行间样式的函数。

解析：

```

function getStyle(obj,attr,value){
    if(!value){
        if(obj.currentStyle){
            return obj.currentStyle(attr);
        } else{
            obj.getComputedStyle(attr,false);
        }
    } else{
        obj.style[attr]=value;
    }
}

```

第 121 道：写一个函数来验证 E-mail 的有效性。

解析：

```

<html>
    <head>
        <meta http-equiv="content-type" content="text/html;
charset=GB2312" />
    </head>
    <body>
        <input type="text" id="email"/>
        <input type="button" value=" 提交 " onclick="doSubmit()"/>
        <script type="text/JavaScript">
            // 来自 JavaScriptCN
            var email = document.getElementById('email');
            function doSubmit() {
                var emailText = email.value;
                // 输入是否为空
                if(emailText!=null && emailText!='') {
                    // 长度不能小于 6 位

```

```

        if(emailText.length>=6) {
            // 使用 @ 分割成数组
            var temp = emailText.split('@');
            if(temp.length==2) {
                // 包含且只包含一个 @ 符号
                // 获取 @ 符号后面的部分（即域名）
                var domain = temp[1];
                var lastChar = domain.substring(domain.
                    length - 1, domain.length);
                // 最后一个字符不是数字（非数字的转换结果为 NaN）
                if(parseInt(lastChar).toString()=='NaN'){
                    // 域名包含 . 才是正确的
                    if(domain.indexOf('.')!=-1) {
                        var darr = domain.split('.');
                        var name = darr[0].toUpperCase();
                        alert('您好，来自 ' + name + ' 的
                            用户。');
                        return;
                    }
                }
            }
        }
    }
    alert(' 请输入正确的 E-mail 地址！ ');
}
</script>
</body>
</html>

```

第 122 道：以下哪些是 JavaScript 的全局函数（ ）（多选）

A. escape B. parseFloat C. eval D. setTimeout E. alert

答案：ABCE

2.6 令人抑郁的考题（对象）

第 123 道：foo 对象有 att 属性，那么获取 att 属性的值，以下哪些做法是可以的（ ）（多选）

A. foo.att B. foo("att") C. foo["att"] / foo.style["att"]
D. foo{"att"} E. foo["a"+"t"+"t"]

答案：ACE

解析：

通过 `Object.params` 直接获取对象的属性值。

第 124 道：以下哪条语句会产生错误（ ）（多选）

- A. `var obj=()`; B. `var obj={}'` C. `var obj=//;`
D. `var obj=[];` E. `var obj=new Object();`

答案：ABC

解析：

对象创建方式：

`objectName = {property1:value1,property2:value2,...,propertyN:valueN}`

其中，`property` 是对象的属性；

`value` 则是对象的值，值可以是字符串、数字或对象三者之一。

构造函数方式：

编写一个构造函数，并通过 `new` 方式来创建对象，构造函数本可以带有构造参数。

```
例如：function User(name,age){
    this.name=name;
    this.age=age;
    this.canFly=false;
}
var use=new User();
```

私有属性定义：

私有属性只能在构造函数内部定义与使用。

语法格式：`var propertyName=value;`

实例属性定义存在两种方式。

(1) `prototype` 方式。

语法格式：`functionName.prototype.propertyName=value`

(2) `this` 方式。

语法格式：`this.propertyName=value`。

注意后面例子中 `this` 使用的位置。

上面 `value` 可以是字符串、数字和对象。

第 125 道：已知对象 `var obj = {.....}`，但对象的属性未知，如何对该对象的属性进行遍历？

解析：

```
function allPrpos(obj) {
    // 用来保存所有的属性名称和值
    var props = "";
    // 开始遍历
```

```

for(var p in obj){
    // 方法
    if(typeof(obj[p])=="function"){
        obj[p]();
    }else{
        // p 为属性名称, obj[p] 为对应属性的值
        props+= p + "=" + obj[p] + "\t";
    }
}
// 最后显示所有的属性
alert(props);
}

```

第 126 道：在 JavaScript 中，如何实现对象的私有属性，并说明原理。

解析：

在 JavaScript 中没有块级作用域的概念，同样也没有私有属性的概念，但是存在私有变量。如果我们想把一些数据封装隐藏起来要怎么做呢？想必大家已经想到了，可以通过使用闭包 + 私有变量的方式来实现对象的私有属性。

1. 实例私有属性

实例私有属性的特点就是每个对象都会包含独立的属性，对象和对象之间没有共享。为了实现这个目标，可以在构造函数中增加一个私有变量，然后定义公共方法来访问这个私有变量。

2. 静态私有属性

在有些情况下，我们可能希望数据全局共享，那么就会用到静态属性，同时，我们还希望这个属性为私有的，那么怎样实现静态私有属性呢？首先这个私有应该在构造函数的外部，为了把构造函数外部的变量和构造函数结合为一体，可以使用闭包把私有变量和构造函数都包含在其作用域中，为了在闭包外面访问内部的构造函数，可以使用一个全局变量来引用构造函数。

第 127 道：请使用 JavaScript 语言创建一个对象来代表一个学生，学生主要有以下属性：姓名 Jeriy（字符串类型）/ 年龄 22（整型）/ 三个朋友（Li、Chen、Zhang，数组）/ 会踢足球（类型为方法，弹出 'football' 字符串即可），并调用 Jeriy 踢足球（弹出 football 字符）。

解析：

```

function xs(name,age,frIEnd){
    this.name=name;
    this.age=age;
    this.frIEnd=frIEnd;
}
xs.prototype.play=function(){
    alert("football")
}

```



```

    }
    var s = new xs("Jeriy",22,["li","chen","zhang"]);
    s.play();

```

第 128 道：在 JavaScript 中，obj.childNodes() 和 obj.children() 的区别是什么？

解析：

(1) childNodes：标准属性，它返回指定元素的子元素集合，包括 HTML 节点、所有属性、文本节点。

可以通过 nodeType 来判断是哪种类型的节点，只有当 nodeType==1 时才是元素节点，值为 2 时是属性节点，值为 3 时是文本节点。

(2) children：非标准属性，它返回指定元素的子元素集合。

但它只返回 HTML 节点，甚至不返回文本节点，虽然不是标准的 DOM 属性，但它和 innerHTML 方法一样，得到了几乎所有浏览器的支持。

因此，如果想获取指定元素的第一个 HTML 节点，可以使用 children[0] 来替代 getFirst 函数。

第 129 道：编写一个函数，去掉数组的重复元素。

解析：

```

function unique(arr) {
    var result = [], isRepeated;
    for (var i = 0, len = arr.length; i < len; i++) {
        isRepeated = false;
        for (var j = 0, len = result.length; j < len; j++) {
            if (arr[i] == result[j]) {
                isRepeated = true;
                break;
            }
        }
        if (!isRepeated) {
            result.push(arr[i]);
        }
    }
    return result;
}

```

第 130 道：如何创建一个对象？请举例说明。

解析：

我们可以利用 JavaScript 的语法特征，以类的思想来创建对象。

(1) 原始方法，代码如下。

```

<script type="text/javascript">
    var obj = new Object();

```

```

obj.name = "Koji"; // 为对象添加属性
obj.age = 21;
obj.showName = function(){ // 为对象添加方法
    alert(this.name);
}
obj.showAge = function(){
    alert(this.age);
}
obj.showName(); //Koji
obj.showAge(); //21
</script>

```

上面的方式通过 **new** 关键字生成一个对象，然后根据 JavaScript 是动态语言的特性来添加属性和方法，构造一个对象。其中的 **this** 表示调用该方法的对象。

这种方式的问题是如果需要多次创建对象，那么需要重复代码多次，不利于代码的复用。

(2) 工厂方法，代码如下。

```

<script type="text/javascript">
    function createObj(){
        var obj = new Object(); // 创建对象
        obj.name = "Koji";
        obj.age = 21;
        obj.showName = function(){
            alert(this.name);
        }
        obj.showAge = function(){
            alert(this.age);
        }

        return obj; // 返回对象
    }

    var obj1 = createObj();
    var obj2 = createObj();

    obj1.showName(); //Koji
    obj2.showAge(); //21
</script>

```

下面这种方式提高了代码重用率，还可以改变工厂方法，传入参数赋值。

代码如下：

```

<script type="text/javascript">
    function createObj(name, age){ // 构造对象时可以传入初始化参数

```

```

    var obj = new Object(); // 创建对象
    obj.name = name;
    obj.age = age;
    obj.showName = function(){
        alert(this.name);
    }
    obj.showAge = function(){
        alert(this.age);
    }
    return obj; // 返回对象
}
var obj1 = createObj("Koji", 22);
var obj2 = createObj("Luo", 21);
obj1.showName(); //Koji
obj1.showAge(); //22
obj2.showName(); //Luo
obj2.showAge(); //21
</script>

```

该方式虽然可以提高代码的重用率，但和面向对象中类的概念相比，有一个很大的缺陷。面向对象强调对象的属性私有，但对象的方法是共享的。而上面的工厂方法在创建对象时，要为每个对象创建各自私有的方法。同时，由于为每个对象都创建逻辑相同的方法，所以很浪费内存。

改进代码如下：

```

<script type="text/javascript">
    function createObj(name, age){
        var obj = new Object(); // 创建对象
        obj.name = name;
        obj.age = age;
        obj.showName = showName;
        obj.showAge = showAge;
        return obj; // 返回对象
    }
    function showName(){ // 函数也是一个对象
        alert(this.name);
    }

    function showAge(){
        alert(this.age);
    }
    var obj1 = createObj("Koji", 22);
    var obj2 = createObj("Luo", 21);
    obj1.showName(); //Koji

```

```
obj1.showAge(); //22
obj2.showName(); //Luo
obj2.showAge(); //21
```

```
</script>
```

上面通过定义几个函数对象，解决了不同对象持有函数对象的私有问题。现在所有对象的方法都持有上面两个函数的引用。但这么一来，对象的函数又和对象相互独立了，这和面向对象中特定方法属于特定类的思想不符合。

(3) 构造函数方式，代码如下：

```
<script type="text/javascript">
    // 定义一个构造函数，用来生成对应的对象，可以类比 Java 中的构造函数
    function Person(name, age) {
        // 当调用 new Person 时，在执行第一行代码前，先生成一个 Person 对象，
        // 并将对象在内存中的索引赋值给 this 关键字，此时可以通过 this 关键字
        // 操作新生成的对象，如下面的添加属性或方法
        this.name = name; //this 关键字不能少。为当前对象，即 this 关键字引用的
        // 对象的 name 属性赋值实际相当于为当前对象添加 name 属性后，再为其
        // name 属性赋值
        this.age = age;
        this.showName = function() { // 为当前对象添加方法
            alert(this.name);
        }
        this.showAge = function() {
            alert(this.age);
        }
        // 将当前对象返回给赋值符号左边的变量（不必明确使用 return）
    }
    var obj1 = new Person("Koji", 22); // 生成一个 Person 对象
    var obj2 = new Person("Luo", 21);
    obj1.showName(); //Koji
    obj1.showAge(); //22
    obj2.showName(); //Luo
    obj2.showAge(); //21

</script>
```

构造函数的方式和工厂方式一样，会为每个对象创建独享的函数对象。当然也可以将这些函数对象定义在构造函数外面，这样又有了对象和方法相互独立的问题。

(4) 原型方法，代码如下：

```
<script type="text/javascript">
    function Person() {} // 定义一个空构造函数，且不能传递参数
```

```

        // 将所有的属性的方法都赋予 prototype 属性
        Person.prototype.name = "Koji"; // 添加属性
        Person.prototype.age = 22;
        Person.prototype.showName = function(){ // 添加方法
            alert(this.name);
        }
        Person.prototype.showAge = function(){
            alert(this.age);
        }
        var obj1 = new Person(); // 生成一个 Person 对象
        var obj2 = new Person();
        obj1.showName(); //Koji
        obj1.showAge(); //22
        obj2.showName(); //Koji
        obj2.showAge(); //22
    </script>

```

当生成 Person 对象时，prototype 的属性都赋值给了新的对象。那么属性和方法是共享的。首先，该方法的问题是构造函数不能传递参数，每个新生成的对象都有默认值。其次，方法共享没有任何问题，但是，当属性是可改变状态的对象时，属性共享就有问题了。

代码如下：

```

<script type="text/javascript">
    function Person(){} // 定义一个空构造函数，且不能传递参数
    Person.prototype.age = 22;
    Person.prototype.array = new Array("Koji", "Luo");
    Person.prototype.showAge = function(){
        alert(this.age);
    }
    Person.prototype.showArray = function(){
        alert(this.array);
    }
    var obj1 = new Person(); // 生成一个 Person 对象
    var obj2 = new Person();
    obj1.array.push("Kyo"); // 向 obj1 的 array 属性添加一个元素
    obj1.showArray(); //Koji,Luo,Kyo
    obj2.showArray(); //Koji,Luo,Kyo
</script>

```

上面的代码通过 obj1 向 obj1 的属性 array 添加元素时，obj2 的 array 属性的元素也跟着受到影响，原因就在于 obj1 和 obj2 对象的 array 属性引用的是同一个 Array 对象，那么改变这个 Array 对象，另一引用该 Array 对象的属性自然也会受到影响，混合的构造函数 / 原型方式使用构造函数定义对象的属性，使用原

型 (prototype) 定义对象的方法, 这样就可以做到属性私有, 而方法共享。

(5) 混合的构造函数 / 原型方式, 代码如下。

```
<script type="text/javascript">
    function Person(name, age) {
        this.name = name;
        this.age = age;
        this.array = new Array("Koji", "Luo");
    }
    Person.prototype.showName = function() {
        alert(this.name);
    }
    Person.prototype.showArray = function() {
        alert(this.array);
    }
    var obj1 = new Person("Koji", 22); // 生成一个 Person 对象
    var obj2 = new Person("Luo", 21);
    obj1.array.push("Kyo"); // 向 obj1 的 array 属性添加一个元素
    obj1.showArray(); // Koji, Luo, Kyo
    obj1.showName(); // Koji
    obj2.showArray(); // Koji, Luo
    obj2.showName(); // Luo
</script>
```

属性私有后, 改变各自的属性不会影响别的对象。同时, 方法也是由各个对象共享的。在语义上, 这符合了面向对象编程的要求。

(6) 动态原型方法, 代码如下:

```
<script type="text/javascript">
    function Person(name, age) {
        this.name = name;
        this.age = age;
        this.array = new Array("Koji", "Luo");
        // 如果 Person 对象中的 _initialized 为 undefined, 表明还没有为 Person 的原型添加方法
        if (typeof Person._initialized == "undefined")
        {
            Person.prototype.showName = function() {
                alert(this.name);
            }

            Person.prototype.showArray = function() {
                alert(this.array);
            }
        }
    }
    Person._initialized = true;
</script>
```

```

    }

    Person._initialized = true; // 设置为 true，不必再
    为 prototype 添加方法
  }
}

var obj1 = new Person("Koji", 22); // 生成一个 Person 对象
var obj2 = new Person("Luo", 21);
obj1.array.push("Kyo"); // 向 obj1 的 array 属性添加一个元素
obj1.showArray(); // Koji, Luo, Kyo
obj1.showName(); // Koji
obj2.showArray(); // Koji, Luo
obj2.showName(); // Luo
</script>

```

这种方法和构造函数 / 原型方式大同小异。只是将方法的添加放到了构造函数之中，同时在构造函数 `Person` 上添加了一个属性用来保证 `if` 语句只能成功执行一次。在实际应用中，采用最广泛的是构造函数 / 原型方法。动态原型方法也很流行，它在功能上和构造函数 / 原型方法是等价的。不要单独使用构造函数 / 原型方法。

第 131 道：以下哪个不是 JavaScript 的内置对象（ ）

A. Time B. Array C. Date D. Math

答案：A

解析：

内置对象是指由 ECMAScript 规范定义的对象或者类。例如，数据、函数、日期等。

JavaScript 有 11 种内置对象：Array、String、Date、Math、Boolean、Number、Function、Global、Error、RegExp、Object。

第 132 道：什么是 JS 的原型模型及原型链？

解析：

原型模型的主要思想是，先借用已有系统作为原型模型，通过不断改进“样品”，使得最后的产品就是用户所需要的。原型模型通过向用户提供原型来获取用户的反馈，使开发出的软件能够真正反映用户的需求。同时，原型模型采用逐步求精的方法完善原型，使得原型能够快速开发，避免了像瀑布模型一样在冗长的开发过程中难以对用户的反馈做出快速的响应。相对瀑布模型而言，原型模型更符合人们开发软件的习惯，是目前较流行的一种实用软件生存期模型。

原型链一般在定义构造函数时用到，可以认为是针对构造函数的或者说是

针对构造函数对应的类的。JavaScript 没有对应继承的关键字，所以用原型链来模拟继承的效果。

2.7 夜以继日难以攻克的考题（定时器）

第 133 道：注释的代码是否可以实现？如不能实现请修改。

```
function test(){
    this.name="taobao";
    this.waitMes=function(){
        // 隔 5 秒钟执行 alert (this.name)
    }
}
```

解析：

不能实现，修改如下：

```
function test(){
    this.name="taobao";
    this.waitMes=function(){
        var that=this;
        setInterval(function(){
            alert(that.name);
        },5000);
    };
}
```

第 134 道：请选择以下代码输出的值（ ）

```
console.log(1);
setTimeout (function(){
    console.log(2);
},0);
setTimeout (function(){
    console.log(3);
},0);
console.log(4);
```

A. 1,2,3,4 B. 1,3,2,4 C. 1,4,2,3 D. 1,4,3,2

答案：C

解析：

JavaScript 代码是按顺序执行的，所以先输出 1。setTimeout (function(){f},0) ,0 的作用很简单，就是为了把 f 放到运行队列的最后去执行。也就是说，无论 setTimeout(function(){f},0) 写在哪，都可以保证在队列的最后执行。因此，结果就是 1,4,2,3。

第 135 道： `setInterval(function() { setTimeout(function() { alert('say!') }, 5000) }, 1000)`; 代码运行后，多久提示一次？

解析：

进入页面 6 秒后弹出 “say!”，之后 1 秒弹出一 “say!”。

先碰到 `setInterval` 间歇定时器 1 秒后执行，再碰到 `setTimeout` 延时定时器 5 秒后执行。可想而知，第一次在 6 秒后弹出，`setTimeout()` 从载入后延迟指定的时间去执行一个表达式或者函数，仅执行一次。`setTimeout()` 只执行 code 一次，1 秒弹出一。如果要多次调用，请使用 `setInterval()` 或者让 code 自身再次调用 `setTimeout()`。

第 136 道： 关于 `setTimeout("check",10)` 中说法正确的是（ ）

- A. 程序循环执行 10 次
- B. Check 函数每 10 毫秒执行一次
- C. 10 作为参数传给函数 check
- D. Check 函数每 10 秒执行一次

答案： B

解析：

`setTimeout`：延迟执行一次定时器上的时间是以毫秒为单位的。

第 137 道： 请描述 `setTimeout` 和 `setInterval` 的区别？在性能上谁更好？为什么要用 `setTimeout` 和 `setInterval` ？

解析：

因为 `setTimeout`（表达式，延时时间）在执行时，是在载入后的延迟指定时间去执行一次表达式，记住，次数是一次；而 `setInterval`（表达式，交互时间）则不一样，它从载入后，每隔指定的时间就执行一次表达式。所以，两者完全是不一样的。很多人习惯将 `setTimeout` 包含于被执行函数中，然后在函数外再次使用 `setTimeout` 来达到定时执行的目的。这样，函数外的 `setTimeout` 在执行函数时再次触发 `setTimeout`，从而形成周而复始的定时效果，使用时各有各的优势。使用 `setInterval`，需要手动停止 tick 触发。而使用嵌套 `setTimeout`，可以根据方法内部本身的逻辑不再调用 `setTimeout`，此时就等于停止了触发。其实两种方法完全可以相互模拟，具体使用哪个，就要看当时的需要而定了。

第 138 道： 下面三个 `alert` 的执行顺序是怎么样子的？结果是什么？

```
function test(){
  var a=1;
  setTimeout(function(){
    alert(a);
    a=3;
  },1000);
  a=2;
  setTimeout(function(){
    alert(a);
```

```

        a=4
    }, 3000)
}
test();
alert(0)

```

答案：结果是 0 2 3

解析：

首先 setTimeout 是一个异步延迟函数，此时先弹 0 是很明显的，因为 test 中的两个 alert() 都是延迟的，在弹 0 之前 test 中的 a 是 2，0 之后就是第一个定时器的函数 alert(a)，此时弹 2，原因在上面解释了，在弹 2 之后执行了 a=3。然后是第二个定时器，所以弹 3。

2.8 学无止境（日期和时间）

第 139 道：按照格式 ×××× 年 ×× 月 ×× 日 ×× 时 ×× 分 ×× 秒 动态显示时间，要求不满 10 的在前面补 0。

解析：

```

<script type="text/javascript"><!--
    new function() {
        with(new Date())
        {
            var t=function(a){return a<10?"0"+a:a;}
            alert(getFullYear()+" 年 "+t(getMonth()+1)+" 月
"+t(getDate())+" 日 "+t(getHours())+" 时 "+t(getMinutes())+" 分
"+t(getSeconds())+" 秒");
        }
    }
// --></script>

```

第 140 道：输出明天的日期。

解析：

```

<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
<title>JS 获取明天日期 </title>
</head>

<body>
<script language="javascript" type="text/javascript">

```

```
function GetDateStr(AddDayCount) {
    var dd = new Date();
    dd.setDate(dd.getDate()+AddDayCount); // 获取 AddDayCount
    天后的日期
    var y = dd.getFullYear();
    var m = dd.getMonth()+1; // 获取当前月份

    日期
    var d = dd.getDate();
    return y+"-"+m+"-"+d;
}
document.write("<br />明天: "+GetDateStr(1));
</script>

</body>
</html>
```

第 141 道：编写一个 JavaScript 函数，实时显示当前时间，格式“-年-月-日时：分：秒”。

解析：

```
<script type="text/javascript">
    function getTime() {
        var now_Date = new Date();
        var nowYear;
        var nowMonth;
        var nowDay;
        var nowdate;
        var nowHourse;
        var nowMinutes;
        var nowSeconds;
        nowYear = now_Date.getYear();
        nowMonth = now_Date.getMonth() + 1;
        nowDay = now_Date.getDay();
        if(nowDay == 0){
            nowDay = "日";
        }

        nowdate= now_Date.getDate();
        nowHourse = now_Date.getHours();
        if(nowHourse == 0){
            nowHourse = 12;
        }
        nowMinutes = now_Date.getMinutes();
        nowSeconds = now_Date.getSeconds();
```

```

        var timerDetails = '今天是:'+nowYear+'年'+nowMonth+'
        月'+nowdate+'日'+ ' ' +'星'+nowDay+ ' ' +'nowHourse+'
        时'+nowMinutes+'分'+nowSeconds+'秒'+ ' ' ';
        alert(timerDetails);
    }
</script>

```

2.9 主流技术的“最爱”（typeof、instanceof）

第 142 道：在 JavaScript 中 typeof 返回的结果有哪几种？

解析：

JavaScript 里面的 typeof 有 6 种：object、function、string、boolean、number、undefined。

第 143 道：下列 JavaScript 代码执行后，依次 alert 的结果是 _____。

```

(function test(){
    var a=b=5;
    alert(typeof a);
    alert(typeof b);
})()
alert(typeof a);
alert(typeof b)

```

答案：number number undefined undefined

解析：

作用域问题，外边的访问不到里面的。所以，外边的会出 undefined。

第 144 道：写出下面的输出结果，并说出原因。

```

typeof("string")
typeof(new String("string"))
typeof(Array)
typeof(new Array())
typeof("s123")
Typeof("123s")

```

解析：

typeof 可以用来检测给定变量的数据类型，可能的返回值：

- (1) 'undefined' —— 这个值未定义。
- (2) 'boolean' —— 这个值是布尔值。
- (3) 'string' —— 这个值是字符串。
- (4) 'number' —— 这个值是数值。
- (5) 'object' —— 这个值是对象或 null。

(6) 'function' ——这个值是函数。

- `typeof("string")` 返回类型就是 `string` 类型；
- `typeof(new String("string"))` 返回类型是 `object`。`new String("string")` 创建一个 `String` 对象，其语法为 `newString(stringValue)`，这个调用会将参数转换为字符串，并作为一个 `String` 对象；
- `typeof(Array)` 返回类型是 `object`；
- `typeof(new Array())` 返回类型是 `object`；
- `typeof("s123")` 返回类型是 `string`；
- `typeof("123s")` 返回类型是 `string`。

第 145 道：简述 `typeof` 和 `instanceof` 的作用。

解析：

`typeof` 返回一个字符串，用于说明元算数的类型；

`instanceof` 判断一个变量是否是某个对象（类）的实例，返回值是布尔类型的。

第3章



进得武林，入得四方 [JavaScript 高级面试题]

想要在武林中站稳脚跟，毫无疑问你需要多方面发展，“对称数”使你应对自如，“继承和多态”使你出神入化，“表单”让你意气风发，“this”应用博大精深，这个阶段会给你带来更强的战斗力。

3.1 轻松解决缠绕你的考题（this）

第 146 道：this 和 a 是什么？

```
(function(a) {  
    console.log(this);  
    return a;  
}).apply(0, [4, 3]);
```

解析：

这是 this 的作用域的问题，这里的 this 并不指向函数，而是指向调用它的主体对象，a 指的是接收的参数。

第 147 道：(function(global, undefined){
 //code
})(this)

说说这段代码中，this 和 undefined 的作用。

解析：

因为 ECMAScript 是从里到外执行 JavaScript 代码的，因此把全局变量 window 或 jQuery 对象传进来，就避免了到外层去寻找，从而提高了效率。undefined 在老一辈的浏览器是不被支持的，直接使用会报错，JavaScript 框架要考虑到兼容性，因此增加一个形参：undefined。

还有，不要用 window.undefined 传递给形参，因为有可能 window.undefined 被其他人修改了，所以最好就是什么都不传，此时形参的 undefined 就是真正的 undefined 了。

第 148 道：简述 JavaScript 中对 this 的理解，并举例说明（至少两个场景）

解析：

this 代表函数运行时，自动生成的一个内部对象，只能在函数内部使用。

比如

```
function test(){
    this.x = 1;
}
```

随着函数使用场合的不同，`this` 的值会发生变化。但是有一个总的原则，那就是 `this` 指的是调用函数的那个对象。

下面分四种情况，详细讨论 `this` 的用法。

情况一：纯粹的函数调用。

这是函数的最通常用法，属于全局性调用，因此 `this` 就代表全局对象 `Global`。

请看下面这段代码，它的运行结果是 1。

```
function test(){
    this.x = 1;
    alert(this.x);
}
test(); // 1
```

为了证明 `this` 就是全局对象，这里对代码做一些改变：

```
var x = 1;
function test(){
    alert(this.x);
}
test(); // 1
```

运行结果还是 1。再变一下：

```
var x = 1;
function test(){
    this.x = 0;
}
test();
alert(x); // 0
```

情况二：作为对象方法的调用。

函数还可以作为某个对象的方法调用，这时 `this` 就指这个上级对象。

```
function test(){
    alert(this.x);
}
var o = {};
o.x = 1;
o.m = test;
o.m(); // 1
```

情况三：作为构造函数调用。

所谓构造函数，就是通过这个函数生成一个新对象（object）。这时，`this` 就

指这个新对象。

```
function test(){
    this.x = 1;
}
var o = new test();
alert(o.x); // 1
```

运行结果仍然为 1。为了表明这时 this 不是全局对象，这里对代码做一些改变：

```
var x = 2;
function test(){
    this.x = 1;
}
var o = new test();
alert(x); //2
```

运行结果为 2，表明全局变量 x 的值根本没变。

情况四：apply 调用。

apply() 是函数对象的一个方法，它的作用是改变函数的调用对象，它的第一个参数就表示改变后的调用这个函数的对象。因此，this 指的就是第一个参数。

```
var x = 0;
function test(){
    alert(this.x);
}
var o={};
o.x = 1;
o.m = test;
o.m.apply(); //0
```

当 apply() 的参数为空时，默认调用全局对象。因此，这时的运行结果为 0，证明 this 指的是全局对象。如果把最后一行代码修改为 o.m.apply(o); //1，运行结果就变成了 1，证明了这时 this 代表的是对象 o。

3.2 细心可以拿满分的题（事件）

第 149 道：在下拉菜单中，用户更改表单元素 Select 中的值时，就会调用（ ）事件处理程序。

A. onChange B. onFocus C. onMouseOver D. onClick

答案：A

解析：

JavaScript 中 onChange 事件是在客户端改变输入控件的值。

第 150 道：（ ）事件处理程序可用于在用户单击按钮时执行函数。

A. onSubmit B. onChange C. onClick D. onBlur

答案：C

解析：

事件就是用户或浏览器自身执行的某种动作，如 click、load、mouseover 都是事件的名称。

第 151 道：下列哪项是按下键盘事件（ ）（多选）

A. onKeyDown B. onKeyPress C. keyCode D. onMouseOver

答案：AB

解析：

(1) keydown()。

keydown 事件会在按下键盘时触发。

(2) keypress()。

keypress 事件会在敲击按键时触发，我们可以理解为按下并抬起同一个按键。

第 152 道：写一段 JavaScript，实现监听页面上所有 a 标签的 click 事件。

解析：

原生的写法：

```
document.getElementsByTagName('a').onclick = function () {}
```

用 jQuery 的话就这么写：

```
$('#a').click(function(){
});
```

第 153 道：IE 和标准 DOM 的事件模型有什么不同？

解析：

IE 内核的浏览器事件模型是冒泡型事件。也就是说，在 IE 内核下，事件句柄的触发顺序是从 ChildNode 到 ParentNode。

```
<div id="ancestor">
  <button id="child">
    Open the console and click me
  </button>
</div>
```

以上的 HTML 代码在 IE 内核下，事件是这样传播的：

```
Button#child;
div#ancestor;
Body;
Document
```

切记！IE 的内核是没有捕获事件过程的，那么在 DOM 标准的浏览器中，事件传播的过程又是怎样的呢？我们依然拿上面的 HTML 代码作为例子，在 DOM 标准的浏览器事件中是这样传播的：

```
Window
Document
Body
div#ancestor
Button#child
```

```
捕获事件
div#ancestor
Body
Document
Window
```

DOM 标准的浏览器中多了一个事件捕获过程，也就是说，当开发者在一个元素上注册了事件后，这个事件的响应顺序是从 window（顶层）开始一级一级地向下传播，然后到了该元素后事件捕获过程结束，事件开始冒泡，一级一级向父层元素冒泡（请注意第 6 步）。当然了，开发者可以很轻松地决定 DOM 标准的浏览器中的事件需要在哪个传播过程触发，这就需要谈一下事件的注册机制。

DOM 标准的浏览器事件是通过 addEventListener 方法注册的，而 IE 内核的浏览器则是通过 attachEvent 方法注册的。

第 154 道：当单击按钮时，如何实现两个 id 的值互换。

解析：

例如：<html>

```
<head>
<script>
    function subOnClick(){
        var txt1=document.getElementById("txt1").value;
// 取 txt1 文本
        var txt2=document.getElementById("txt2").value;
// 取 txt2 文本
        // 开始互换
        document.getElementById("txt1").value=txt2;
        document.getElementById("txt2").value=txt1;
    }
    window.onload=function(){
        document.getElementById("submitBtn1").onclick=
subOnClick;
        document.getElementById("submitBtn2").onclick=
subOnClick;
    }
</script>
```

```

</head>

<body>
    <input type="text" value="12345666" id="txt1" />
    <input type="submit" id="submitBtn1" />
    <input type="text" value="12345222" id="txt2" />
    <input type="submit" id="submitBtn2" />
</body>
</html>

```

第 155 道：怎样使用事件？IE 和 DOM 事件模型之间存在哪些主要差别？

解析：

使用事件：

(1) 当我们要阻止浏览器中某个 DOM 元素的默认行为时，在 W3C 标准里调用 `e.preventDefault()`，而在 IE 下则通过设置 `window.event.returnValue=false` 来实现。

(2) 当我们要阻止冒泡事件时，在 W3C 标准里调用 `e.stopPropagation()`，而在 IE 下通过设置 `window.event.cancelBubble=true` 来实现。

差别：

(1) Traditional Module。

传统方式的事件模型即直接在 DOM 元素上绑定事件处理器，例如：

```

window.onload=function() {…}
obj.onmouseover=function(e) {…}
obj.onclick=function() {…}

```

首先这种方式无论在 IE 还是 Firefox 等其他浏览器上，都可以成功运行，这便是它最大的优势，而且在 Event 处理函数内部的 `this` 变量无一例外都被绑定到当前 DOM 元素，这使得 JavaScript 程序员可以大大利用 `this` 关键字做很多事情。至于它的缺点也很明显，就是传统方式只支持 Bubbling，而不支持 Capturing，并且在 DOM 元素上一次只能绑定一个事件处理器，无法实现多 Handler 绑定，最后就是 `function` 参数中的 `event` 参数只对非 IE 浏览器有效果（因为 IE 有 `window.event`）。

(2) W3C (Firefox.e.g) Event Module。

Firefox 等浏览器很坚决地遵循 W3C 标准来制定浏览器事件模型，使用 `addEventListener` 和 `removeEventListener` 两个函数，看几个例子：

```

indow.addEventListener("load",function() {…},false);
document.body.addEventListener("keypress",function{…},false);
obj.addEventListener("mouseover",MV,true);
function MV() {…}

```

`addEventListener` 带有三个参数，第一个参数是事件类型，就是我们熟知的

那些事件名字去掉前面的“on”，第二个参数是处理函数，可以直接给函数自变量或者函数名，第三个参数是 boolean 值，表示事件是否支持 Capturing。

W3C 的事件模型优点是 Bubbling 和 Capturing 都支持，并且可以在一个 DOM 元素上绑定多个事件处理器，各自并不会冲突。并且在处理函数内部时，this 关键字仍然可以使用只想被绑定的 DOM 元素。另外 function 参数列表的第一个位置（不管是否显示调用）永远都是 event 对象的引用。

至于它的缺点，很不幸地就是在市场份额最大的 IE 浏览器下不可使用这一点。

（3）IE Event Module。

IE 自己的事件模型跟 W3C 类似，但主要是通过 attachEvent 和 detachEvent 两个函数来实现的。依旧看几个例子：

```
window.attachEvent("onload",function(){...});
document.body.attachEvent("onkeypress",myKeyHandler);
```

可以发现，它跟 W3C 的区别是没有第三个参数，而且第一个表示事件类型的参数也必须把“on”加上。

这种方式的优点就是在同一个 DOM 元素上能绑定多个事件处理函数。

至于它的缺点，为什么如今在实际开发中很少见呢？首先 IE 浏览器本身只支持 Bubbling，不支持 Capturing；而且在事件处理的 function 内部，this 关键字也无法使用，因为 this 永远都只想得到 window object 这个全局对象。要想得到 event 对象，必须通过 window.event 方式；最后一点，在别的浏览器中，它显然是无法工作的。

第 156 道：IE 和 Firefox 事件对象怎么处理兼容性？

解析：

如果在使用 JavaScript 时涉及 event 处理，就需要知道 event 在不同的浏览器中的差异，主要的 JavaScript 事件模型有三种，它们分别是 NN4、IE4+ 和 W3C/Safar。

1. window.event

【分析说明】先看一段代码：

```
function et(){
    alert(event);//IE: [object]
}
```

以上代码在 IE 中运行的结果是 [object]，而在 Firefox 中无法运行。

因为在 IE 中 event 作为 window 对象的一个属性可以直接使用，但是在 Firefox 中却使用了 W3C 的模型，它是通过传参的方法来传播事件的，也就是说，你需要为你的函数提供一个事件响应的接口。

【兼容处理】添加对 event 的判断，根据浏览器的不同来得到正确的 event。

```
function et(){
    evt=evt?evt:(window.event?window.event:null);
```

```
// 兼容 IE 和 Firefox
```

```
alert(evt);
```

```
}
```

2. 键盘值的取得

【分析说明】IE 和 Firefox 获取键盘值的方法不同，可以理解为，Firefox 下的 event.which 与 IE 下的 event.keyCode 相当。

```
function myKeyPress(evt) {
    // 兼容 IE 和 Firefox 获得 keyBoardEvent 对象
    evt = (evt) ? evt : ((window.event) ? window.event : "");
    // 兼容 IE 和 Firefox 获得 keyBoardEvent 对象的键值
    var key = evt.keyCode?evt.keyCode:evt.which;
    if(evt.ctrlKey && (key == 13 || key == 10)){
        // 同时按下 Ctrl+Enter 组合键
        //do something;
    }
}
```

3. 事件源的获取

【分析说明】在使用事件委托时，通过事件源获取来判断事件到底来自哪个元素，但是，在 IE 下，event 对象有 srcElement 属性，却没有 target 属性；在 Firefox 下，event 对象有 target 属性，却没有 srcElement 属性。

【兼容处理】

```
ele=function(evt){ // 捕获当前事件作用的对象
    evt=evt||window.event;
    return
    (obj=event.srcElement?event.srcElement:event.target);
}
```

4. 事件监听

【分析说明】在事件监听处理方面，IE 提供了 attachEvent 和 detachEvent 两个接口，而 Firefox 提供的是 addEventListener 和 removeEventListener。

【兼容处理】最简单的兼容性处理就是封装这两套接口。

```
function addEvent(elem, eventName, handler) {
    if (elem.attachEvent) {
        elem.attachEvent("on" + eventName, function(){
            handler.call(elem)});
        // 此处使用回调函数 call(), 让 this 指向 elem
    } else if (elem.addEventListener) {
        elem.addEventListener(eventName, handler, false);
    }
}

function removeEvent(elem, eventName, handler) {
```

```

if (elem.detachEvent) {
    elem.detachEvent("on" + eventName, function() {
        handler.call(elem) });
    // 此处使用回调函数 call(), 让 this 指向 elem
} else if (elem.removeEventListener) {
    elem.removeEventListener(eventName, handler,
        false);
}
}

```

需要特别注意, 在 Firefox 下, 事件处理函数中的 this 指向被监听元素本身, 而在 IE 下则不然, 可使用回调函数 call, 让当前上下文指向监听的元素。

5. 鼠标位置

【分析说明】在 IE 下, event 对象有 x、y 属性, 但是没有 pageX、pageY 属性; 在 Firefox 下, event 对象有 pageX、pageY 属性, 但是没有 x、y 属性。

【兼容处理】使用 mX(mX = event.x ? event.x : event.pageX;) 来代替 IE 下的 event.x 或者 Firefox 下的 event.pageX。复杂点还要考虑绝对位置。

```

function getAbsPoint(e) {
    var x = e.offsetLeft, y = e.offsetTop;
    while (e = e.offsetParent) {
        x += e.offsetLeft;
        y += e.offsetTop;
    }
    alert("x:" + x + "," + "y:" + y);
}

```

第 157 道: 如何为元素绑定多个事件, 要求同时支持 Firefox 和 IE。

解析:

```
event = event || window.event; // 使 event 兼容 Firefox 与 IE
```

Firefox 的 Firebug, 不仅能测试 JavaScript, 还能检查 CSS 错误, 是一般常用的。但它主要检查 Firefox 方面的错误, 对 IE 就无能为力了。

要测试 IE, 就用 IETester, 它几乎可以测试 IE 所有版本 (1.0 版本恐怕也用不着测试了), 用法也很方便。至于 JavaScript 对不同浏览器的兼容注意事项, 的确很多, 下面给出的仅仅只是一部分。

一般建议还是采用 jQuery、prototype 等一些已经处理好了兼容的脚本库, 更重要的是, 它们简化了很多操作, 还提供了平常你很难实现的增强功能。

JavaScript 兼容浏览器 Firefox 和 IE 技巧:

做 B/S 开发难免会用到 JavaScript, 而每个浏览器对 JavaScript 的支持有所不同, 这就需要程序员去兼容它们, 不然有些浏览器就无法运行我们的代码。

第 158 道: 请写一个通用的事件侦听器函数。

解析：

关于 JavaScript 中的事件监听大家用得比较多了，无非是判断浏览器是否支持 `addEventListener` 和 `attachEvent`，网上搜索关于事件监听的方法也很多，但是总有些不是很完善。下面的方法中对于添加事件监听的方法是一样的，只不过在取消事件绑定上做了点“手术”，现在可以支持匿名函数的使用，所以在绑定事件时不再需要给函数单独命名了。

主要代码如下：

```

    /* 绑定事件与取消绑定 */
    var handleHash = {};
    var bind = (function() {
        if (window.addEventListener) {
            return function(el, type, fn, capture) {
                el.addEventListener(type, function(){
                    fn();
                    handleHash[type] = handleHash[type] || [];
                    handleHash[type].push(arguments.callee);
                }, capture);
            };
        } else if (window.attachEvent) {
            return function(el, type, fn, capture) {
                el.attachEvent("on" + type, function(){
                    fn();
                    handleHash[type] = handleHash[type] || [];
                    handleHash[type].push(arguments.callee);
                });
            };
        }
    })();
    var unbind = (function(){
        if (window.addEventListener) {
            return function(el, type) {
                if (handleHash[type]){
                    var i = 0, len = handleHash[type].length;
                    for (i; i<len ; i += 1){
                        el.removeEventListener(type,
                            handleHash[type][i]);
                    }
                }
            };
        } else if (window.attachEvent) {
            return function(el, type) {

```

```

        if (handleHash[type]){
            var i = 0, len = handleHash[type].length;
            for (i; i<len ; i += 1){
                el.detachEvent("on" + type,
                    handleHash[type][i]);
            }
        };
    };
}
})();

```

原理解析：

handleHash 用作哈希表缓存事件的 function，handleHash['事件名称'] 是一个数组，用来添加多个事件监听的方法，当需要移除哪个事件时，遍历 handleHash['事件名称'] 的数组，然后移除。

使用方法：

```

bind(obj, 'click', function(){
    alert ('click');
});
unbind(obj, 'click');

```

第 159 道：当浏览器窗体大小发生变化时，哪个事件将会被触发？

解析：

当浏览器的窗口大小被改变时，触发的事件 window.onresize 为事件指定代码：window.onresize = function(){}。

例如浏览器可见区域信息如下：

```

<span id="info_keleyi_com"> 请改变浏览器窗口大小 </span>
<script>
window.onresize = function(){
    document.getElementById("info_ke"+"leyi_com").innerHTML=" 宽度: "+document.documentElement.clientWidth+", 高度: "+document.documentElement.clientHeight;
}
</script>

```

第 160 道：简述 JavaScript 的事件模型。

解析：

1. 原始事件模型

属性事件处理模式如下。

(1) 基本事件处理。其实大多数人使用的 JavaScript 事件处理模式都是这种代码方式。

(2) 事件类型。分为“输入事件（如 onclick）”和“语义事件（如 onsubmit）”。

2. 标准事件模型

DOM2 对其作了标准化。

(1) 先由 document 向目标对象传播，称之为捕捉阶段。

(2) 目标对象的事件处理程序运行。

(3) 从目标对象向 document 起泡。

3. IE 事件模型 (IE5.5/IE6)

(1) 传播过程只起泡，不捕捉。起泡中断方法：window.event.cancelBulle=true。

(2) event 对象不是事件处理程序的函数参数，而是 window 的全局变量。

(3) 事件注册函数 attachEvent() 和反注册函数 detachEvent()。

4. Netscape 事件模型

由于 Netscape 已经完全停止开发，所以就不详述了，简单地说，就是只捕捉不冒泡。

第 161 道：JavaScript 中 mouseover 与 mouseenter，mouseout 与 mouseleave 的区别。

解析：

mouseover 与 mouseenter 的区别：

- 不论鼠标指针穿过被选元素或其子元素，都会触发 mouseover 事件。
- 只有在鼠标指针穿过被选元素时，才会触发 mouseenter 事件。

mouseout 与 mouseleave 的区别：

- 不论鼠标指针离开被选元素还是任何子元素，都会触发 mouseout 事件。
- 只有在鼠标指针离开被选元素时，才会触发 mouseleave 事件。

第 162 道：简述 addEventListener 和 attachEvent 的作用，两者是否有区别？

解析：

1. 支持的浏览器

(1) addEventListener 在支持 DOM2 的浏览器中使用，如 Firefox、Chrome 等。

(2) attachEvent 为 IE 所用。

2. 处理程序执行阶段

(1) addEventListener 的第三个参数为 true 时，在捕获阶段执行；为 false 时，在冒泡阶段执行。

(2) attachEvent 均在冒泡阶段执行。

3. 作用域

(1) addEventListener 的作用域为元素作用域，this 为 element 引用。

(2) addEvent 的作用域为全局作用域，this 为 window 引用。

4. 事件处理程序执行顺序

(1) addEventHandler：执行顺序与添加顺序一致。

(2) attachEvent: 执行顺序与添加顺序相反。

第 163 道: document.load 和 document.ready 有何区别?

解析:

页面加载完成有两种事件, 一是 ready, 表示文档结构已经加载完成 (不包含图片等非文字媒体文件); 二是 onload, 表示页面包含图片等文件在内的所有元素都加载完成。

第 164 道: 讲述 JavaScript 的事件绑定的方法, 并举例说明。

解析:

1. 在 DOM 中直接绑定

例如:

```
<input onclick="alert(' 谢谢支持 ')"type="button" value=" 单击我, 弹出警告框"
">
```

2. 在 JavaScript 代码中绑定

例如, 为 id="demo" 的按钮绑定一个事件, 显示它的 type 属性。

```
<input id="demo" type="button" value=" 单击我, 显示 type 属性 ">
```

```
<script type="text/javascript">
    document.getElementById("demo").onclick=function(){
        alert(this.getAttribute("type")); // this 指当前发生
        事件的 HTML 元素, 这里是 <div> 标签
    }
</script>
```

3. 绑定事件监听函数

绑定事件的另一种方法是用 addEventListener() 或 attachEvent() 来绑定事件监听函数。

3.3 不可忽视的小漏洞 (表单、文本框)

第 165 道: 下列选项中, 描述正确的是 () (多选)

- A. options.add(new Option("a","A")) 可以动态添加一个下拉列表选项
- B. option.add(new Option("a","A")) 可以动态添加一个下拉列表选项
- C. new Option("a","A") 中 "a" 表示列表选项的值, "A" 用于在页面中显示
- D. new Option("a","A") 中 "A" 表示列表选项的值, "a" 用于在页面中显示

答案: AD

解析:

```
new option (text,value,defaultSelected,selected);
```

- text: 字符串, 指定 option 对象的 text 属性, 即 <option></option> 之间的文字;
- value: 字符串, 指定 option 对象的 value 属性;
- defaultSelected: 布尔值, 指定 option 对象的 defaultSelected 属性;
- selected: 布尔值, 指定 option 对象的 selected 属性。

第 166 道: JavaScript 进行表单验证的目的是 ()

- A. 把用户的正确信息提交给服务器
- B. 检查提交的数据必须符合实际
- C. 使得页面变得美观、大方
- D. 减轻服务器端的压力

答案: B

解析:

JavaScript 可用在数据被送往服务器前对 HTML 表单中的这些输入数据进行验证。

第 167 道: 如何限制多行文本框内容长度在 10 以内?

解析:

```
<textarea name="txt1" cols="45" rows="2" onKeyDown='if
(this.value.length>=10){event.returnValue=false}'></textarea>
```

第 168 道: 如何获取下拉框中选中项的内容?

解析:

```
<select id="test" name="">
    <option value="1">text1</option>
    <option value="2">text2</option>
</select>
```

code:

1. JavaScript 原生的方法

- 拿到 select 对象: Var myselect=document.getElementById("test");
 - 拿到选中项的索引: var index=myselect.selectedIndex// selectedIndex 代表的是你所选中项的 index;
 - 拿到选中项 options 的 value: myselect.options[index].value;
 - 拿到选中项 options 的 text: myselect.options[index].text。
2. jQuery 方法 (前提是已经加载了 jQuery 库)
- var options=\$("#test option:selected") // 获取选中的项;
 - alert(options.val()) // 拿到选中项的值;
 - alert(options.text()) // 拿到选中项的文本。

第 169 道: 在 IE9、Opera、Chrome 和 Safari 浏览器中, 在用户选择了文本并释放鼠标的情况下触发 () 事件

A. blur B. focus C. select D. change

答案: C

解析:

textarea 和 input 文本框都支持 select() 方法, 这个方法用于选择文本框的所有文本。

与 select() 方法对应的是一个 select 事件。在选择了文本框时, 就会触发 select 事件。不过, 到底什么时候触发 select 事件, 还因浏览器而异。在 IE9、Opera、Firefox、Chrome 和 Safari 中, 只有用户选择了文本 (而且要释放鼠标), 才会触发 select 事件。而在 IE8 及更早版本中, 只要用户选择了一个字母 (不必释放鼠标), 就会触发 select 事件。另外, 在调用 select() 方法时也会触发 select 事件。

第 170 道: table 标签中 border、cellpadding, 以及 td 标签中 colspan、rowspan 分别起什么作用?

解析:

- border: 边界;
- cellpadding: 边距;
- colspan: 跨列数;
- rowspan: 跨行数。

第 171 道: 在 Form 中的 input 可以设置 readonly 和 disable, 请问这两项属性有什么区别?

解析:

`<input name="country" id="country" size=12 value="disabled">` 提交时得不到该值 " disabled="disabled" > 放在 Form 表单中提交后得不到该值。将 disabled="disabled" 改为 readonly="readonly" 即可。

设置为 disabled 的 input 将会有下面的限制: 不能接收焦点, 使用 Tab 键时将被跳过, 被限制的对象值将不会被传递到后台程序。

设置为 readonly 的 input 将会有下面的限制: 可以接收焦点, 但不能被修改, 可以使用 Tab 键进行导航, 只有 successful 的表单元素才是有效数据, 才可以进行提交。被限制的对象值与后台交互时值才传递。

第 172 道: 表单中的输入域 input 和文本框 textarea 的异同?

解析:

相同点——input 标签和 textarea 标签的使用目的都是让用户或站长来提交数据, 为了让另一方看到想要的的数据资料, 特别说明的是提交后处理数据都是一样的。

区别:

(1) input 标签只能设置单行文本, input 标签填写格式是单独出现的, 而

textarea 可设置多行文本，带滚动条的标签填写格式是成对出现的。

(2) input 标签编写代码时大多用来放置字数较少的单行文字内容，而 textarea 一般让用户可以输入多行文字，输入的文字信息量比较大。

(3) input 标签的 value 值可以直接写在元素里面，而 textarea 标签的 value 值必须写在开始标签和结束标签之间。

第 173 道：如何获得表单 <select> 域的选择部分的文本？

解析：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=gb2312"
  />
    <title> 如何获取表单 <select> 域的选择部分的文本 zgbn</title>
  </head>

  <script type="text/JavaScript">
    function test() {
      var select0 = document.getElementById("select");

      var s = select0.value ; // 获取已经选择的值

      var i = select0.selectedIndex ;// 获取选择第几个选项的索引

      var op = select0.options ;// 获取所有选项的集合

      var t = op.text ;// 获取指定索引的集合中元素的文本

      alert(t) ;
    }
  </script>

  <body>
    <select name="select" id="select">
      <option value=1>11</option>
      <option value=2>22</option>
      <option value=3>33</option>
      <option value=4>44</option>
      <option value=5>55</option>
```

```

        </select>

        <input type="button" value=" 单击 " onclick=
"test()" />

    </body>
</html>

<html>
    <head>
        <title></title>
        <script>
function sel(obj){
alert(" 显示文本: "+obj.options[obj.selectedIndex].text);
        alert(" 值: "+obj.options[obj.
selectedIndex].value);
    }
    </script>
</head>
<body>
        <form name="a">
            <select name="a" size="1" onchange=
"sel(this)">

                <option value="a" >1</option>
                <option value="b" >2</option>
                <option value="c" >3</option>
            </select>

        </form>
    </body>
</html>

```

第 174 道：在 Form 表单中 get 与 post 方式提交的区别。

解析：

(1) get 是从服务器上获取数据，post 是向服务器传送数据。

(2) get 是把参数数据队列加到提交表单的 action 属性所指的 URL 中，值和表单内各个字段一一对应，在 URL 中可以看到。post 是通过 HTTP post 机制，将表单内各个字段与其内容放置在 HTML HEADER 内一起传送到 action 属性所指的 URL 地址。用户看不到这个过程。

(3) get 传送的数据量较小，不能大于 2KB。post 传送的数据量较大，一般默认为不受限制。但理论上，IIS4 中最大值为 80KB，IIS5 中最大值为 100KB。

(4) get 安全性非常低，post 安全性较高。

(5) get 限制 Form 表单的数据集的值必须为 ASCII 字符，不是 ASCII 字符时要转换为 ASCII 字符，也就是“%XX”（其中 XX 为该符号以 16 进制数表示的 ASCII 或 ISO Latin-1 值）这种；而 post 支持整个 ISO10646 字符集。

第 175 道：使用 JavaScript 输出下拉列表中所有的选项的文本。

```
<form>
请选择您喜欢的水果
<select id="mySelect">
    <option> 苹果 </option>
    <option> 桃子 </option>
    <option> 香蕉 </option>
    <option> 橘子 </option>
</select>
<br/><br/>
<input type="button" onclick="getOption()" value="输出所有选项"/>
</form>
```

解析：

```
<script type="text/javascript">
    function getOptions()
    {
        var x=document.getElementById("mySelect");
        var y="";
        for (i=0;i<x.length;i++)
        {
            y+=x.options[i].text;
            y+="<br />";
        }
        document.write(y);
    }
</script>
```

option 集合可返回包含 <select> 元素中所有 <option> 的一个数组。

注释：数组中的每个元素对应一个 <option> 标签，由 0 起始。语法为：selectObject.options[]。

第 176 道：写一个简单的 Form 表单，当光标离开表单时把表单的值发送给后台。

解析：

```
<script type="text/javascript">

    $("#o_form").blur(function(){
        var vals=$(this).val();
        $.Ajax({
```

```

        url:"./index.php",           // 传给后台的地址
        type:"post",                 // 传输方式
        data:{$val:vals},           // 传输数据
        success: function (data) { // 如果执行成功，那么执行
            此方法
                alert(data);          // 用 data 来获取后台传过
            来的数据，或者是单纯的语句
        }
    })
});
</script>

```

第 177 道：在 Form 中的 input 有哪些类型？

解析：

类型有：text、radio、checkbox、file、button、image、submit、reset、hidden。

submit 是 button 的一个特例，也是 button 的一种，它把提交这个动作自动集成了。

如果表单在单击提交按钮后需要用 JavaScript 进行处理（包括输入验证）后再提交的话，那么必须把 submit 改成 button，即取消其自动提交的行为，否则，将会造成提交两次的结果，对于动态网页来说，也就是对数据库操作两次。

button 具有 name、value 属性，能触发 onclick 事件。

submit 继承了 button。

button 提交的是 innerText。

submit 增加了触发表单 onsubmit 事件的功能、增加了执行表单的 submit() 方法的功能。

input type=submit，按回车提交表单。

第 178 道：当单击按钮时，如何实现两个 td 值的互换？用 JavaScript 实现此功能。

解析：

```

<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=gb2312" />
        <title> </title>
        <script type="text/JavaScript">
            //<![CDATA[
                /*

```

分析：

这道题首先考变量传值，其次考如何取元素的值。

代码如下：


```

*/
function submitbtn()
{
    var tText1 = document.getElementById('txt1');
    var SubmitBtn1 = document.getElementById('submitBtn1');
    var tText2 = document.getElementById('txt2');
    var SubmitBtn2 = document.getElementById('submitBtn2');

    SubmitBtn1.onclick = function()
    {
        var temp = tText1.value;
        tText1.value = tText2.value;
        tText2.value = temp;
    };
    SubmitBtn2.onclick = function()
    {
        var temp = tText2.value;
        tText2.value = tText1.value;
        tText1.value = temp;
    };
}

window.onload = function()// 窗口加载的时候
{
    submitbtn();
}
</script>

</head>
<body>
    <input type="text" value="12345666" id="txt1" />
    <input type="button" id="submitBtn1" />
    <input type="text" value="12345222" id="txt2" />
    <input type="button" id="submitBtn2" />
</body>
</html>

```

第 179 道：希望图片具有“提交”按钮的功能，该如何编写表单提交（ ）

- A. 在图片的 onClick 事件中手动提交
- B. 在图片上添加 onSubmit 事件
- C. 在图片的 onSubmit 事件中手动提交

D. 在表单中自动提交

答案: B

解析:

首先用图片代替提交按钮 submit 和重置按钮 reset, 然后给图片加 onclick 事件。

```
<script language="javascript">
    function fsubmit(obj){
        obj.submit();
    }
    function freset(obj){
        obj.reset();
    }
</script>
<form id="form1" name="form1" method="post" action="login.asp">
    姓名 <input type="text" name="textfIEld" />
    
    
</form>
```

第 180 道: 创建一个文本框, 让其宽度为 120 像素, 高度为 20 像素, 对齐方式为居中对齐, 写出该代码。

解析:

```
<textarea name="" id="" cols="30" rows="10"
style="width:120px;
height: 20px; text-align: center;"></textarea>
```

第 181 道: 如何用 JavaScript 获取所有选中的 checkbox 的值?

解析:

```
// 说明: 用 JavaScript 验证表单 (Form) 中多选框 (checkbox) 的值
function getCheckboxValue(checkbox) {

    if (!checkbox.length && checkbox.type.toLowerCase() ==
'checkbox')
    { return (checkbox.checked)?checkbox.value:''; }
    if (checkbox[0].tagName.toLowerCase() != 'input' || chec
        kbox[0].type.toLowerCase() != 'checkbox')
    { return ''; }
    var val = [];
    var len = checkbox.length;
    for(i=0; i<len; i++)
    {
```

```

        if (checkbox[i].checked)
        {
            val[val.length] = checkbox[i].value;
        }
    }
    return (val.length)?val:'';
}

```

注意：上面的代码传入的参数是 checkbox 对象，如果页面报错，那么可以删除代码中的第一个 if。

3.4 “照镜子”看题（对称数）

第 182 道：一个数字倒着读时，和原数字相同，我们将这个数字称之为对称数，例如（1,121,88,8998），在不考虑性能的情况下，请找出 1 ~ 10000 之间的对称数，要求用 JavaScript 实现。

解析：

```

<script>
    for(var i=1;i<=10000;i++){
        var str_i=i.toString(),l=str_i.length; // 将数字转化为字符串，获取字符串的长度
        var arr_i=str_i.split(""); // 将字符串转化为数组
        var rev_arr=[]; // 命名一个空数组
        for(var j=0;j<l;j++){ // 遍历1~10000之间的所有数字
            rev_arr.unshift(arr_i[j]); // 实现数组反转
        }

        var rev_str=rev_arr.join(""); // 将数组转化为数字
        if(str_i==rev_str){ // 将原来的数字与反转后的数字作比较
            document.write(str_i+"<br>"); // 如果是相等的，就是对称数，并返回
        }
    }
</script>

```

3.5 让你坚定不移看下去（JavaScript 客户端检测）

第 183 道：什么是富客户端，什么是薄（瘦）客户端？

解析：

富客户端，英文名为 Rich Client。简单地讲，非 IE 浏览器的程序一般可以看作胖客户端，IE 浏览器可以看作瘦客户端。详细地讲，可以这么说：一个程序能够通过下载文件来操作、运行一个应用，或者从一个文件服务器请求一个基于应用的服务。它需要安装，并且不同于一个薄客户端（Thin Client），比如一个普通的 Web 页面。富客户端为一个客户端，它有着复杂的 UI 界面和交互。下面用例子来说明什么是富客户端。

- (1) Java Applet，应用于 Web 页面。
- (2) JavaScript Application，比如 Bindows。
- (3) J2ME MIDP Midlets on a phone/PDA。
- (4) Macromedia Application 公司的 Flash 技术。
- (5) 微软公司的 MFC 应用程序。
- (6) Eclipse 的 Rcp 程序。
- (7) Flex 程序（属于 Flash 一类的比较新的技术）。

第 184 道：JavaScript 可以在以下什么地方执行（ ）？

- A. 服务器端的 IE 浏览器 B. 客户端的 IE 浏览器
C. 在服务器端的 Tomcat 容器里 D. 在客户端的 Tomcat 容器里

答案：B

解析：

JavaScript 没有独立的运行窗口，浏览器当前窗口就是它的运行窗口。

3.6 过了这题，公司随便挑（排序）

第 185 道：请根据每个元素的 i 属性，由小到大排序如下数组。

```
var ar=[{i:5,v:1},{i:2,v:4},{i:3,v:2},{i:1,v:5},{i:4,v:3}];
```

解析：

```
ar.sort(function(a, b) {
    5 return a.i - b.i;
    6 })
```

对于一个数组，sort() 默认按字符编码排序：

```
var testArray=[3,324,5345,6546,134,5654,665];
testArray.sort();
alert(testArray);
```

输出结果是：134 3 324 5345 5654 6546 665

现在要让它按照数值大小排序：

```
var testArray=[3,324,5345,6546,134,5654,665];
```

```
testArray.sort(function(a,b){return a-b;});
alert(testArray);
```

这里传递一个比较函数给 sort，比较函数的逻辑是：如果两参数的差值小于 0，则表示 a 必须出现在 b 前面，否则在 b 后面。

输出结果是：3 134 324 665 5345 5654 6546

第 186 道：说说 JavaScript 数组排序方法 sort() 的使用，重点介绍 sort() 参数的使用及其内部机制。

解析：

sort() 方法用于对数组的元素进行排序。

语法：arrayObject.sort(sortby)

参数：sortby

描述：可选。规定排序顺序。必须是函数。

返回值：对数组的引用。请注意，数组在原数组上进行排序，不生成副本。

【说明】

(1) 如果调用该方法时没有使用参数，将按字母顺序对数组中的元素进行排序，说得更精确点，是按照字符编码的顺序进行排序。要实现这一点，首先应把数组的元素都转换成字符串（如有必要），以便进行比较。

(2) 如果想按照其他标准进行排序，就需要提供比较函数，该函数要比较两个值，然后返回一个，用于说明这两个值的相对顺序的数字。比较函数应该具有两个参数：a 和 b，其返回值如下：

若 a 小于 b，在排序后的数组中 a 应该出现在 b 之前，返回一个小于 0 的值。

- 若 a 等于 b，则返回 0；
- 若 a 大于 b，则返回一个大于 0 的值。

3.7 看了这题，收获多多（call、apply）

第 187 道：call 和 apply 的区别。

解析：

对于第一个参数，其意义都一样，但对第二个参数：apply 传入的是一个参数数组，也就是将多个参数组合成为一个数组传入，而 call 则作为 call 的参数传入（从第二个参数开始），如 func.call(func1, var1, var2, var3) 对应的 apply 写法为：

```
func.apply(func1,[var1,var2,var3])
```

同时使用 apply 的好处是可以直接将当前函数的 arguments 对象作为 apply 的第二个参数传入。

第 188 道：call 的作用是什么？它的参数是如何传递的？

解析:

call 方法在 msdn 中的解释为, 调用一个对象的一个方法, 以另一个对象替换当前对象。

JavaScript 中, call 的语法为: call(thisObj,arg1,...,argn)。

- 参数 thisObj: 可选项, 将被用作当前对象的对象。
- 参数 arg1,...,argn: 可选项, 将被传递方法参数序列。

3.8 看懂必高薪的面试题 (继承和多态)

第 189 道: 请用代码说明 JavaScript 如何实现继承和多态。

解析:

1. 继承

(1) 原型继承法。

```
function parentClass(){
    //private fIEld
    var x = "I'm a parentClass fIEld!";
    //private method
    function method1() {
        alert(x);
        alert("I'm a parentClass method!");
    }
    // public fIEld
    this.x = "I'm a parentClass object fIEld!";
    // public method
    this.method1 = function() {
        alert(x);
        alert(this.x);
        method1();
    }
}

parentClass.prototype.method = function () {
    alert("I'm a parentClass prototype method!");
}

parentClass.staticMethod = function () {
    alert("I'm a parentClass static method!");
}
```

(2) 调用继承法。

```
function parentClass() {
    // private fIEld
```

```

var x = "I'm a parentClass fIEld!";
// private method
function method1() {
    alert(x);
    alert("I'm a parentClass method!");
}
// public fIEld
this.x = "I'm a parentClass object fIEld!";
// public method
this.method1 = function() {
    alert(x);
    alert(this.x);
    method1();
}
}
parentClass.prototype.method = function () {
    alert("I'm a parentClass prototype method!");
}
parentClass.staticMethod = function () {
    alert("I'm a parentClass static method!");
}
}

```

2. 多态

(1) 重载。

```

function add(){
    var sum=0;
    for(var i=0;i<arguments.length;i++){
        sum+=arguments[i];
    }
    return sum;
}

```

(2) 覆盖。

```

function parentClass() {
    this.method = function() {
        alert("parentClass method");
    }
}
function subClass() {
    this.method = function() {
        alert("subClass method");
    }
}
subClass.prototype = new parentClass();

```

```
subClass.prototype.constructor = subClass;

var o = new subClass();
o.method();
```

3.9 大型企业面试必考（charAt()、indexOf()）

第 190 道：讲述 JavaScript 函数中 charAt() 和 indexOf() 的区别。

解析：

charAt() 语法：strObj.charAt(index)

返回指定索引位置处的字符；

indexOf() 语法：strObj.indexOf(subString[,startIndex])

返回 String 对象内第一次出现子字符串的字符位置。

3.10 五年前端，三年必考（substr、substring）

第 191 道：JavaScript 中 substr 和 substring 的区别是什么？

解析：

stringvar.substr(start[,length]) 返回一个从指定位置开始的指定长度的子字符串。如果 length 为 0 或负数，将返回一个空字符串。如果没有指定该参数，则子字符串将延续到 stringvar 的最后。

strVariable.substring(start, end) 返回位于 String 对象中指定位置的子字符串。返回一个包含从开始到最后（不包含 end）的字符串。

第 192 道：用 JavaScript 取 www.qdjhu.com/public/images/test.jpg 字符串 test.jpg 的扩展名。

解析：

用 url=www.qdjhu.com/public/images/test.jpg 存储网址，m=url.lastIndexOf(".") 存储最后一个 "." 在字符串中的索引，fileName=url.substr(m+1) 得到 "jpg"。

3.11 这题如此而已，我会为你加油（iframe）

第 193 道：关于 iframe 表述正确的有（ ）（多选）

- A. 通过 iframe，网页可以嵌入其他网页内容，并可以动态更改
- B. 在相同域名下，内嵌的 iframe 可以获取外层网页的对象

- C. 在相同域名下，外层网页脚本可以获取 iframe 网页内的对象
- D. 可以通过脚本调整 iframe 的大小

答案：ABCD

解析：

iframe 元素也就是文档中的文档，或者好像浮动的框架（frame）。通过 iframe 对象所在页面的对象模型，用户可以访问 iframe 对象的属性，但不能访问其内容。所以本题正确答案为 ABCD。

第 4 章



一手遮天，大名远扬 [JavaScript 终极面试题]

一路走来不容易，在这最后时刻，恭喜你，你已成功修炼了一本秘籍，可以在 JavaScript 界大名远扬了。Ajax 独霸一方，无刷新更是让你闻风丧胆，逃之夭夭。

4.1 让人暴走的考题（Ajax）

第 194 道：Ajax 是什么？它的全称是？

解析：

Ajax 的全称为 “Asynchronous JavaScript and XML”（异步 JavaScript 和 XML），它是一种创建交互式网页应用的网页开发技术。

第 195 道：简述 Ajax 中 JavaScript 脚本缓存问题，如何解决？

解析：

修改 JavaScript 内容，调试时并不能显示新写的代码的结果，是因为 JavaScript 为了加速页面执行，当前页面会使用缓存来保持当前调用的相同链接。

解决方法：为了开发时调试方便，可以在链接地址的后面增加一个随机函数。

第 196 道：Ajax 和 JavaScript 的区别？

解析：

JavaScript 是一种在浏览器端执行的脚本语言；Ajax 是一种创建交互式网页应用的开发技术，它利用了一系列相关的技术，其中就包括 JavaScript。

JavaScript 是由网景公司开发的一种脚本语言，它和 Sun 公司的 Java 语言是没有任何关系的，它们相似的名称只是一种营销策略。在一般的 Web 开发中，JavaScript 是在浏览器端执行的，我们可以用 JavaScript 控制浏览器的行为和内容。在 Ajax 应用中，信息在浏览器和服务端之间是传递是通过 XML 数据或者字符串实现的。

第 197 道：Ajax 应用和传统 Web 应用有什么不同？

解析：

在传统的 JavaScript 编程中，如果想得到服务器端数据库或文件上的信息，或者发送客户端信息到服务器，需要建立一个 HTML form，然后 get 或者 post 数据到服务器端。用户需要单击 “submit” 按钮来发送或者接收数据信息，然

后等待服务器响应请求，页面重新加载。因为服务器每次都会返回一个新的页面，所以传统的 Web 应用有可能很慢，而且交互不太好。使用 Ajax 技术，就可以使 JavaScript 通过 XMLHttpRequest 对象直接与服务器进行交互。通过 HTTP Request，一个 Web 页面可以发送一个请求到 Web 服务器，并且接收 Web 服务器返回的信息（不用重新加载页面），展示给用户的还是同一个页面，用户感觉页面刷新了，但看不到 JavaScript 后台进行的发送请求和接受响应。

第 198 道：Ajax 有哪些优点和缺点？

解析：

优点：

- (1) 最大的优点是页面无刷新，用户的体验非常好。
- (2) 使用异步方式与服务器通信，具有更加迅速的响应能力。
- (3) 可以把以前一些服务器负担的工作转嫁到客户端，利用客户端闲置的能力来处理，减轻服务器和带宽的负担，节约空间和宽带租用成本。并且减轻服务器的负担，Ajax 的原则是“按需取数据”，可以最大程度地减少冗余请求和响应对服务器造成的负担。

(4) 基于标准化的并被广泛支持的技术，不需要下载插件或者小程序。

缺点：

- (1) Ajax 不支持浏览器返回按钮。
- (2) 安全问题，Ajax 暴露了与服务器交互的细节。
- (3) 对搜索引擎的支持比较弱。
- (4) 破坏了程序的异常机制。
- (5) 不容易调试。

第 199 道：主要的 Ajax 框架都有什么？

解析：

Dojo(dojotoolkit.org);

Prototype 和 Scriptaculous (www.prototypeJS.org 和 script.aculo.us);

Direct Web Reporting (fetahead.org/dwr);

Yahoo! User Interface Library (developer.yahoo.com/yui);

Google Web Toolkit (code.google.com/webtoolkit);

jQuery。

第 200 道：如何处理 Ajax 的中文乱码问题？

解析：

有中文乱码是因为 JavaScript 页面和 action 类中使用的编码方式不一致造成的，可采用如下两种方式解决：

- (1) 页面的 JavaScript 做两次 encodeURIComponent，服务器的 servlet 获取后做一次 UTF-8 转码，因为前两次进行编码后都变成了英文的字节码，所以到服务器端

无论如何解码都不会错误，推荐使用该方法。

- JavaScript 的代码 `var url="Ajaxserver?name="+encodeURIComponent(encodeURIComponent($("#userName").val()));`

- 服务器端的代码

```
String old = httpServletRequest.getParameter("name");
String name = URLDecoder.decode(old, "UTF-8");
```

(2) 在客户端 JavaScript 做一次编码，在服务器端做一次 ISO-8859-1 和 UTF-8 转换。

- JavaScript 代码

```
var url = "Ajaxserver?name=" + encodeURIComponent($("#userName").val());
```

- 服务器端的代码

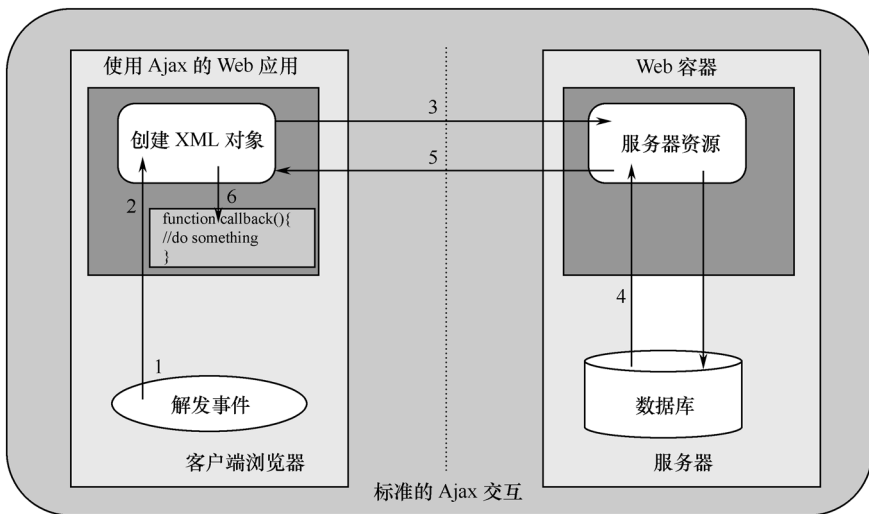
```
String old = httpServletRequest.getParameter("name");
String name = new String(old.getBytes("ISO-8859-1"), "UTF-8");
```

此处采用了硬编码，在个别浏览器中会出现问题，不建议使用。

第 201 道：Ajax 的交互模型（流程）是什么样的？

解析：

Ajax 的交互模型如下图所示。



Ajax 的交互流程如下。

- (1) 启动获取 XMLHttpRequest 对象。
- (2) open 打 URL 通道，并设置异步传输。
- (3) 发送数据到服务器。

(4) 服务器接受数据并处理，处理完成后返回结果。

(5) 客户端接收服务器端返回。

第 202 道：简述一下 Ajax 的工作原理。

解析：

通过 XMLHttpRequest 对象来向服务器发送异步请求，从服务器获得数据，然后用 JavaScript 来操作 DOM，从而更新页面。这其中最关键的一步就是从服务器获得请求数据。

第 203 道：Ajax 跨域的解决办法？

解析：

(1) Web 代理的方式。即用户访问 A 网站时所产生的对 B 网站的跨域访问请求均提交到 A 网站的指定页面，由该页面代替用户页面完成交互，从而返回合适的结果。此方案可以解决现阶段所能够想到的多数跨域访问问题，但要求 A 网站提供 Web 代理的支持，因此 A 网站与 B 网站之间必须是紧密协作的，且每次交互过程，A 网站的服务器负担增加，无法代用户保存 session 状态。

(2) on-Demand 方式。MYMSN 的门户就是用的这种方式，不过 MYMSN 中不涉及跨域访问问题。动态控制 script 标记的生成，通过修改 script 标记的 src 属性完成对跨域页面的调用。此方案存在的缺陷是，script 的 src 属性完成该调用时，采取的是 get 方式，如果请求时传递的字符串过大，程序可能会无法正常运行。不过此方案非常适合聚合类门户使用。

(3) iframe 方式。在 javaeye 上有一篇关于跨域访问的帖子，它提到有人已经用 iframe 的方式解决了跨域访问问题。数据提交和获取，采用 iframe 这种方式的确可以，但由于父窗口与子窗口之间不能交互（跨域访问的情况下，这种交互被拒绝），因此无法完成对父窗口效果的影响。

(4) 用户本地转储方式。IE 本身依附于 Windows 平台的特性为我们提供了一种基于 iframe 利用内存来“绕行”的方案，即两个 Window 之间可以在客户端通过 Windows 剪贴板的方式进行数据传输，只需要在接收数据的一方设置 Interval 进行轮询，获得结果后清除 Interval 即可。Firefox 的平台独立性决定了它不支持剪贴板这种方式，而以往版本的 Firefox 中存在的插件漏洞又被 fixed 了，所以 Firefox 无法通过内存来完成暗度陈仓。而由于文件操作 Firefox 也没有提供支持（无法通过 Cookie 跨域完成数据传递），致使这种技巧性的方式只能在 IE 中使用。

(5) 结合了前面几种方式，在访问 A 网站时，先请求 B 网站完成数据处理，再根据返回的标识来获得所需的结果。这种方法的缺点也很明显，即 B 网站的负载增大了。其优点在于，对 session 实现了保持，同时 A 网站与 B 网站页面间的交互能力增强了。

第 204 道：编写 Ajax 的 4 个主要步骤是什么？

解析：

1. 创建引擎（XMLHttpRequest 对象）。
2. 事件处理函数，处理服务器的响应结果。
3. 打开一个连接：open(method, url, asynch)。
4. 发送数据：send(data)。

第 205 道：请写出一个最熟悉的 JavaScript 框架（架包），以及该框架中 Ajax 的使用方法。

解析：

jQuery

使用方法：jQuery 里的 \$.Ajax 方法的作用是通过 HTTP 请求加载远程数据。该方法是由 jQuery 底层 Ajax 实现的。常用参数 \$.Ajax({type：数据提交方式；url：数据提交路径；data：需要提交的数据；dataType：服务器返回数据的类型；success：请求成功后的回调函数；error：请求失败后的回调函数}) 在回调函数中执行操作。

第 206 道：简述 Ajax 的实现步骤。

解析：

1. 创建 XMLHttpRequest 对象，也就是创建一个异步调用对象。
2. 创建一个新的 HTTP 请求，并指定该 HTTP 请求的方法、URL 及验证信息。
3. 设置响应 HTTP 请求状态变化的函数。
4. 发送 HTTP 请求。
5. 获取异步调用返回的数据。
6. 使用 JavaScript 和 DOM 实现局部刷新。

第 207 道：如何解决 Ajax 的安全性问题。

解析：

增加验证码、增加随机 Token、约束同一请求在规定时间内最大请求数量、服务器端校验数据的正确性、尽量运用 post 办法。

第 208 道：简述 Ajax 异步机制，Ajax 如何实现？

解析：

XMLHttpRequest 对象是 Ajax 技术的核心。XMLHttpRequest 对象使得 JavaScript 脚本能够实现对服务器的异步请求，即向后台发送请求并接收服务器响应，通过动态获取响应数据来更新局部页面。

第 209 道：Ajax 请求时，如何解释 JSON 数据？

解析：

首先写客户端的 HTML 代码。

代码如下：

```
<table>
```

```

<thead>
  <tr>
    <td> 学号 </td>
    <td> 姓名 </td>
    <td> 班别 </td>
    <td> 性别 </td>
    <td> 电话 </td>
  </tr>
</thead>
<tbody></tbody>
</table>
<input id="btnget" type="button" value=" 加载数据 " />

```

其次写 JavaScript 代码。

代码如下：

```

$(function () {
  $("#btnget").click(function () {
    $.Ajax({
      type: "post",
      dataType: "Json",
      url: "data.ashx",
      success: function (msg) {
        var str = "";
        for (i in msg) {
          str += "<tr><td>" + msg[i].id +
            "</td><td>" + msg[i].name +
            "</td><td>" + msg[i].cla +
            "</td><td>" + msg[i].sex +
            "</td><td>" + msg[i].tel +
            "</td></tr>";
        }
        $("tbody").append(str);
      }
    });
  });
});

```

为了使表格好看一些，我们定义一下它的样式。

代码如下：

```

<style type="text/css">
  table {
    border-collapse: collapse;
  }
  table td {

```

```

        text-align: center;
        border: 1px solid gray;
        padding: 3px 10px;
    }
</style>

```

最后写服务器端返回 Json 数据的代码。

代码如下：

```

string data = [{"id\\":\\"2010324268\\",\\"name\\":\\" 林宇 \\",\\"cla\\":\\"10 软件 \\",\\"sex\\":\\" 男 \\",\\"tel\\":\\"13800138000\\"},{\\"id\\":\\"2010324256\\",\\"name\\":\\" 李四 \\",\\"cla\\":\\"10 网络 \\",\\"sex\\":\\" 女 \\",\\"tel\\":\\"10010\\"},{\\"id\\":\\"2010324264\\",\\"name\\":\\" 张三 \\",\\"cla\\":\\"10 软件 \\",\\"sex\\":\\" 男 \\",\\"tel\\":\\"10086\\"}]];
context.Response.Write(data);

```

4.2 面对这些考题，除了崩溃我不知道还能说什么 (XMLHttpRequest 对象)

第 210 道：使用 open 方法打开具有浏览器工具条、地址栏、菜单栏的窗口，下列选项正确的是（ ）

- A. open("x.html", "HI", "toolba=1,scrollbars=1,status=1");
- B. open("HI", "scrollbars=1,location=1,status=1");
- C. open("x.html", "status=yes,menubar=1,location=1");
- D. open("x.html", "HI", "toolba=yes,menubar=1,location=1");

解析：

答案：A

open() 方法用于打开一个新的浏览器窗口或查找一个已命名的窗口。

语法 window.open(url,name,features,replace)

参数含义：

url——一个可选的字符串，声明了要在新窗口中显示文档的 URL。如果省略了这个参数，或者它的值是空字符串，那么新窗口就不会显示任何文档。

name——一个可选的字符串，该字符串是一个由逗号分隔的特征列表，其中包括数字、字母和下划线，该字符声明了新窗口的名称。这个名称可以用作标记 <a> 和 <form> 的属性 target 的值。如果该参数指定了一个已经存在的窗口，那么 open() 方法就不再创建一个新窗口，而只是返回对指定窗口的引用。

features——一个可选的字符串，声明了新窗口要显示的标准浏览器的特征。如果省略该参数，新窗口将具有所有标准特征。在窗口特征这个表格中，我们

对该字符串的格式进行了详细的说明。

replace——一个可选的布尔值。规定了装载到窗口的 URL 是在窗口的浏览历史中创建一个新条目，还是替换浏览历史中的当前条目。支持下面的值：

true，URL 替换浏览历史中的当前条目。**false**，URL 在浏览历史中创建新的条目。

窗口特征：**toolbar=yes|no|1|0** 是否显示浏览器的工具栏。默认是 **yes**。

location=yes|no|1| 是否显示地址字段。默认是 **yes**。

menubar=yes|no|1| 是否显示菜单栏。默认是 **yes**。

第 211 道：介绍一下 XMLHttpRequest 对象的常用方法和属性。

解析：

三个常用的属性如下：

1. **onreadystatechange** 属性（存有处理服务器响应的函数）。
2. **readyState** 属性（存有服务器响应的状态信息）。
3. **responseText** 属性（取回由服务器返回的数据）。

两个方法如下：

1. **open()** 方法。第一个参数定义发送请求所使用的方法，第二个参数规定服务器端脚本的 URL，第三个参数规定应当对请求进行异步处理。

2. **send()** 方法。将请求送往服务器。

第 212 道：XMLHttpRequest 对象在 IE 和 Firefox 中创建方式有没有不同？

解析：

有，该对象在 IE 中通过 **new ActiveXObject()** 得到，而在 Firefox 中通过 **new XMLHttpRequest()** 得到。

第 213 道：HTTP 协议的状态消息有哪些？200、302 的对应描述。

解析：

HTTP 协议的状态消息有：

1xx：信息（100 Continue, 101 Switching Protocols）

2xx：成功（200 OK, 201 Created, 202 Accepted, 203 Non-authoritative Information, 204 No Content, 205 Reset Content, 206 Partial Content）

3xx：重定向（300 Multiple Choices, 301 Moved Permanently, 302 Found, 303 See Other, 304 Not Modified, 305 Use Proxy, 306 Unused, 307 Temporary Redirect）

4xx：客户端错误（400 Bad Request, 401 Unauthorized, 402 Payment Required, 403 Forbidden, 404 Not Found, 405 Method Not Allowed, 406 Not Acceptable, 407 Proxy Authentication Required, 408 Request Timeout, 409 Conflict, 410 Gone, 411 Length Required, 412 Precondition Failed, 413 Request Entity Too Large, 414 Request-url Too Long, 415 Unsupported Media Type, 416 Request

Range not satisfiable, 417 Expectation Failed)

5xx: 服务器错误 (500 Internal Server Error, 501 Not Implemented, 502 Bad Gateway, 503 Service Unavailable, 504 Gateway Timeout, 505 HTTP Version Not Supported)

200 OK: 请求成功 (其后是对 get 和 post 请求的应答文档)。

302 Found: 所请求的页面已经临时转移至新的 URL。

第 214 道: 简述 XMLHttpRequest 对象, 并说明此对象的属性 onreadystatechange 的作用。

解析:

XMLHttpRequest 对象暴露各种属性、方法和事件, 以便于脚本处理和控制 HTTP 请求与响应。XMLHttpRequest 对象提供了对 HTTP 协议的完全的访问, 包括做出 post 和 head 请求, 以及普通的 get 请求的能力。XMLHttpRequest 可以同步或异步地返回 Web 服务器的响应, 并且能够以文本或者一个 DOM 文档的形式返回内容。

onreadystatechange 的作用: 当 readyState 属性发生变化时触发此事件, 用于触发回调函数。

第 215 道: JavaScript 如何得到 HTTP 的请求头信息和返回的头信息?

解析:

getResponseHeader 从响应信息中获取指定的 http 头信息。

语法: strValue= oXMLHttpRequest.getResponseHeader(bstrHeader);

getAllResponseHeaders 获取响应的所有 HTTP 头信息。

语法: strValue=oXMLHttpRequest.getAllResponseHeaders();

第 216 道: HTTP 协议中, 以下 status code 返回值的含义是什么?

302、304、404、500。

解析:

使用 ASP.NET/PHP/JSP 或者 JavaScript 都会用到 HTTP 的不同状态, 一些常见的状态码含义如下。

302 (临时移动): 服务器目前从不同位置的网页响应请求, 但请求者应继续使用原有位置来进行以后的请求。

304 (未修改): 自从上次请求后, 请求的网页未修改过。服务器返回此响应时, 不会返回网页内容。

404 (未找到): 服务器找不到请求的网页。

500 (服务器内部错误): 服务器遇到错误, 无法完成请求。

4.3 感慨“时间太瘦”的考题 (关于继承)

第 217 道: JavaScript 如何实现面向对象中的继承, 请写段简单的代码。

解析：

原型链继承，通过对象 Child 的 prototype 属性指向父对象 Parent 的实例，使 Child 对象实例能通过原型链访问到父对象构造所定义的属性、方法等。

使用 apply、call 方法，由于 JavaScript 内置的 Function 对象的 apply、call 方法改变了对象构造中“this”的上下文环境，使特定的对象实例具有对象构造中所定义的属性、方法。

对象实例间的继承，JavaScript 对象的多态性，允许实例动态地添加属性、方法。该特性造就了 JavaScript 中的另一种继承手法——对象实例间的继承。

```
例：function Parent(){};
      function Child(){};
      Child.prototype = new Parent();
      Child.prototype.constructor = Child;
      var child = new Child();
      alert(child.constructor);//function Parent(){};
      alert(child instanceof Child);//true
```

第 218 道：JavaScript 有哪几种形式可以实现继承，各有什么优缺点？

解析：

(1) 构造继承法：在子类中执行父类的构造函数。

(2) 原型继承法：prototype 的最大特性是能够让对象实例共享原型对象的属性，因此，如果把某个对象作为一个类型的原型，那么我们说这个类型的所有实例都以这个对象为原型。此时，这个对象的类型也可以作为那些以这个对象为原型的实例的类型。

(3) 实例继承法：子类的原型 = 父类的实例。

(4) 复制继承法：复制继承法是通过对象属性的拷贝来实现继承的。

优缺点比较：

比较项	构造继承	原型继承	实例继承	复制继承
静态属性继承	N	Y	Y	Y
内置（核心）对象继承	N	部分	Y	Y
多参多重继承	Y	N	Y	N
执行效率	高	高	高	低
多继承	Y	N	N	Y
instanceof	false	true	false	false

第 219 道：怎样体现 JavaScript 的继承关系？

解析：

由于 JavaScript 是 prototype 的对象模型，所以没有严格意义上的类 class。全部都是对象 Object 实现继承，可以先创建一个父对象。

```
oldObject=function(){
    this.a=" 属性 1"
}
// 复制出一个新对象，新对象里面已经具有旧对象的内容
newObject=new oldObject();
// 新增些内容，扩展新对象
newObject.b=" 属性 2";
newObject.function(){
    // 新方法 1
}
// 新对象具有旧对象的属性
alert (newObject.a);
```

第 220 道：编写一段代码，让 f1 继承 f2 的所有成员。

```
function f2(){
    this.a2=3;
    this.b2=4;
}

function f1(){
    this.a1=1;
    this.b1=2;
}
```

解析：

```
f1.prototype = new f2();
f1.prototype.constructor = f1;
```

第 221 道：选择 JavaScript 的一项特性，例如原型链继承、闭包模型、对象模型，对其进行简单的介绍或者给出一个实例。

解析：

我们经常需要将一个函数对象暂时挂到一个引用上留待后面执行，因为不到执行时是很难知道其具体参数的，而先前将它赋给那个引用时更是不知道。

执行一个与特定的 JavaScript 对象关联的事件处理函数，并且要知道调用该对象的哪个方法。相关的例子是，用 JavaScript 对象来封装与特定 DOM 元素的交互。这个 JavaScript 对象具有 doOnClick、doMouseOver 和 doMouseOut 方法，并且当用户在该特定的 DOM 元素中触发了相应的事件时要执行这些方法。不过，可能会创建与不同的 DOM 元素关联的任意数量的 JavaScript 对象，而且每个对象实例并不知道实例化它们的代码将会如何操纵它们（即注册事件处理函数与定义相应的事件处理函数分离）。这些对象实例并不知道如何在全局环境中引用自身，因为它们不知道引用它们的实例将会指定哪个全局变量（如果有）。

```
<input type="button" name=" 按钮名字 " value="button" id="testbt" />
<script type="text/javascript">
```

```

function associateObjWithEvent(obj, methodName) {
    return (function(e) {
        e = e || window.event;
        return obj[methodName](e, this);
    });
}

function DhtmlObject(elementId) {
    var el = document.getElementById(elementId);
    if(el) {
        el.onmouseover = associateObjWithEvent(this,
            "doMouseOver");
    }
}

DhtmlObject.prototype.doMouseOver = function(event,
element) {
    alert(element.name + " " + event.target + " " +
        event.screenX);
}

var bt = new DhtmlObject("testbt");
//bt();
</script>

```

第 222 道：写一个让 b 继承 a 的方法。

解析：

```

function a(name,age){
    this.age = age ? age : 30;
    this.name = name ? name : '小明'
    this.say = function(){
        alert(this.name + '今年' + this.age + '岁了! ');
    }
}

function b(){
    B.prototype = new a();
    var c = new b();
    c.say();
}

```

第 223 道：JavaScript 实现一个类 A，包含私有属性、公有属性、私有方法和公有方法。

解析：

现在写一个 class1 类：

```
function A(){
    this.name="world";// 公有属性
    var message="No Messages!";// 私有属性
    this.sayHello=function(){// 公有方法（可访问所有权限的方法和属性）
        alert("hello !" + this.name + "I want to say:" + message);
    }
    function getMessage(){// 私有方法（只能访问私有的方法和属性）
        alert(message);
    }
}

class1.staticMethod=function(){// 定义该类的一个静态方法
    alert("staticMethod()");
}
```

说起类，其实 JavaScript 中所有的 function 都可以当作一个类来使用，从上述的例子可以看出，可以 new(实例化) 一个类，也可以直接当作 function 调用。

第 224 道：JavaScript 如何实现继承？请举例说明。

解析：

JavaScript 要实现继承，其实就是实现三层含义：

- (1) 子类的实例可以共享父类的方法。
- (2) 子类可以覆盖父类的方法或者扩展新的方法。
- (3) 子类 and 父类都是子类实例的“类型”。

例子：

使用原型继承，中间使用临时对象作为 Child 的原型属性，临时对象的原型属性再指向父类的原型，防止所有子类和父类原型属性都指向同一个对象，这样当修改子类的原型属性，就不会影响其他子类和父类。

```
function extend(Child, Parent) {
    var F = function(){};
    F.prototype = Parent.prototype;
    Child.prototype = new F();
    Child.prototype.constructor = Child;
    Child.base = Parent.prototype;
}

function Parent(name)
{
    this.aa = 123;
    this.getName = function() {return name;}; // 使用闭包模拟私有成员
    this.setName = function(value){name=value;};
}
```

```

Parent.prototype.print = function(){alert("print!");};
Parent.prototype.hello = function()
{
    alert(this.getName() + "Parent")
};

function Child(name,age)
{
    Parent.apply(this, arguments); // 调用父类构造函数来继承父类
    定义的属性
    this.age = age;
}
extend(Child,Parent); // 继承 Parent

Child.prototype.hello = function() // 重写父类 hello 方法
{
    alert(this.getName() + "Child");
    Parent.prototype.hello.apply(this,arguments); // 调用父类
    同名方法
};
// 子类方法
Child.prototype.doSomething = function(){
    alert(this.age + "Child    doSomething");
};

var p1 = new Child("xhan",22);

var p2 = new Child("xxx",33);

p1.hello();
p2.hello();

p1.doSomething(); // 子类方法
p1.print(); // 父类方法

alert(p1 instanceof Child); //true
alert(p1 instanceof Parent); //true

```

4.4 高级前端必考试题（闭包）

第 225 道：什么是闭包？

解析：

闭包就是能够读取其他函数内部变量的函数。

(1) 闭包外层是一个函数。

(2) 闭包内部都有函数。

(3) 闭包会 `return` 内部函数。

(4) 闭包返回的函数内部不能有 `return` (因为这样就真的结束了)。

(5) 执行闭包后，闭包内部变量会存在，而闭包内部函数的内部变量不会存在。

第 226 道：闭包有什么特性？对页面有什么影响？

解析：

闭包特性：

(1) 作为函数变量的一个引用。当函数返回时，其处于激活状态。

(2) 闭包就是当一个函数返回时，并没有释放资源的栈区。

闭包对页面的影响：

通过使用闭包，我们可以做很多事情。比如模拟面向对象的代码风格；更优雅、更简洁的表达出代码；在某些方面提升代码的执行效率。

第 227 道：闭包的含义是什么？它的作用和原理是什么？具体能在什么场景使用？

解析：

闭包的含义：

闭包就是能够读取其他函数内部变量的函数。

作用和原理：

因为闭包只有在被调用时才执行操作，所以它可以被用来定义控制结构。

多个函数可以使用一个相同的环境，这使得它们可以通过改变那个环境相互交流。

闭包可以用来实现对象系统。

使用的场景：

(1) 采用函数引用方式的 `setTimeout` 调用。

(2) 将函数关联到对象的实例方法。

(3) 封装相关的功能集。

第 228 道：闭包的好处和坏处。

解析：

闭包的好处：

(1) 逻辑连续，当闭包作为另一个函数调用参数时，避免脱离当前逻辑而单独编写额外逻辑。

(2) 方便调用上下文的局部变量。

(3) 加强封装性，是第2点的延伸，可以达到对变量的保护作用。

闭包的坏处：

闭包有一个非常严重的问题，即浪费内存，浪费内存不仅仅因为它常驻内存，更重要的是，对闭包的使用不当会造成无效内存的产生。

第229道：请写出一个闭包的简单实例。

解析：

```
function a(){
    var i=0;
    function b(){
        alert(++i);
    }
    return b;
}
var c=a();
c();
```

这是个标准的闭包。在函数a中，定义了函数b，a又return了b的值。

```
var c=a();
c();
```

这两句执行很重要。var c=a(); 在这里执行了a函数，那么a肯定经过了return，按照主流语言的函数特性，现在c的值就是a的返回值。

第二行c()实际执行的就是b的函数。最后不管执行的是谁，会弹出一个值为0的窗口。到此为止，按理论来说所有的生命周期就算是全部结束了。可是如果我们再多执行一行var c=a(); c();c(); 第一次弹出0，第二次弹出执行的却是1。也就是说第一次c()后，a中的i依然保留，所以a在内存的栈区依然保留。

4.5 值得思考和深钻的考题（video）

第230道：请使用JavaScript创建video标签，并创建播放、暂停方法。

解析：

```
<!DOCTYPE html>
<html>
<head>
    <title>Simple javascript Controller</title>
    <script type="text/javascript">
        function playPause() {
            var myVideo = document.getElementsByTagName('video')[0];
            if (myVideo.paused)
                myVideo.play();
```

```

        else
            myVideo.pause();
        }
        function makeBig() {
            var myVideo = document.getElementsByTagName('video')[0];

            myVideo.height = (myVideo.videoHeight * 2) ;
        }
        function makeNormal() {
            var myVideo = document.getElementsByTagName('video')[0];
            myVideo.height = (myVideo.videoHeight) ;
        }
    }
</script>
</head>
<body>

    <div class="video-player" align="center">
    <video src="myMovIE.m4v" poster="poster.jpg" ></video>
    <br>
    <a href="javascript:playPause();">Play/Pause</a> <br>
    <a href="javascript:makeBig();">2x Size</a> |
    <a href="javascript:makeNormal();">1x Size</a> <br>
    </div>
</body>
</html>

```

4.6 这些年，一直“陪伴我”的考题（url 参数）

第 231 道：请编写一个 JavaScript 函数 `parseQueryString`，它的用途是把 url 参数解析为一个对象，如：

```

var url="http://www.qdjh.com/index.asp?key0=0 & key1=1 & key2=2...";
var obj=parseQueryString(url);
alert(obj.key0); // 输出 0

```

解析：

```

var url="http://www.qdjh.com/index.php?key0=0&key1=1&key2=2";
function parseQueryString(url){
    var str=url.split("?")[1];
    var items=str.split("&");
    var result={}
    var arr=;

```

```

for(var i=0; i<items.length; i++){
    arr=items.split('=');
    result[arr[0]]=arr[1];
}
return result;
var obj=parseQueryString(url);
console.log(obj)

```

第 232 道：获取当前 url，并将 url 的参数遍历数组打印出来，并在 3 秒自动返回前一个页面。

解析：

```

function getUrlPars(){
    var url=location.search;
    var theRequest=new Object();
    if(url.indexOf("?")!=-1){
        var str=url.substr(1),
            strs=str.split("&");
        for(var i=0;i<strs.length;i++){
            var sTemp=strs[i].split("=");
            theRequest[sTemp[0]]=(sTemp[1])

        }
    }
    return theRequest;
}

var mYRequest = GetUrlPars();
var 参数1值 = mYRequest["参数1"];

```

4.7 小心陷阱！总是被坑的考题（JavaScript 模仿块级作用域）

第 233 道：下面 JavaScript 代码的运算结果是 2 还是 undefined？请阐述原因。

```

function show(){
    var b=1;
    a=++b;
}
show();
alert(a);

```

解析：

运算结果是 2。

原因:

因为 $b=1$,

$a=++b$,

所以 $a=1+1=2$ 。

4.8 错误率最高的考题（正则表达式）

第 234 道：请写出一个方法，验证用户输入是否为数字。

解析：

```
<script type="text/javascript">
    function validate(){
        var reg=new RegExp("[0-9]*$");
        var obj=document.getElementById("name");
        if(!reg.test(obj.value)){
            alert("请输入数字!");
            if(!/[0-9]*$/i.test(obj.value)){
                alert("请输入数字!");
            }
        }
    }
</script>
```

第 235 道：写一个匹配手机号的正则表达式。

解析：

$/^{13}[0-9]\{1\}[0-9]\{8\}|^{15}[9]\{1\}[0-9]\{8\}/$

第 236 道：请写出一个正则表达式，验证以下数列：

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

规律是一个数字后面跟一个逗号，最后一个数字后面没有逗号，并且数字只能是 1 到 12 中的任意一个。

解析：

$/^{1}[0-2],|[0-9.])*(1[0-2]|0-9)/$

第 237 道：用正则表达式怎样去掉重复的字符串，而只保留其中的一个？

解析：

```
var str="aaadcccdddd";
str = str.replace(/(.)\1+/g,'$1');
```

第 238 道：用正则表达式判断手机、邮箱、电话、中文。

解析：

手机： $/^{13}[0-9]\{1\}[0-9]\{8\}|^{15}[9]\{1\}[0-9]\{8\}/$

邮箱： $/^{1}[\w-]+(\.[\w-]+)*@[\w-]+(\.[\w-]+)+$/$

电话: /((\d{3})\d{3}-)|(\d{4})\d{4}-)?(\d{8}|\d{7})/

中文: /^[u4e00-\u9fa5]{0,}\$/

4.9 前端最新技术考题（转换大写）

第 239 道：使用 JavaScript 将字符串中由空格隔开的每个单词首字母大写。

解析：

字符串中，每个单词由空格隔开，空格的个数不限。

代码如下：

```
function capitalize(string) {
    var words = string.split(" ");
    for(var i = 0; i < words.length; i++) {
        words[i] = words[i].charAt(0).toUpperCase() +
        words[i].slice(1);
    }
    return words.join(" ");
}
var string = "Ajax Cookie Event object";
capitalize(string); // "Ajax Cookie Event Object"
```

第 240 道：如何把一个字符串里的所有单词的第一个字符转换为大写？

解析：

```
var words = string.split(" ");
for(var i = 0; i < words.length; i++) {
    words[i] = words[i].charAt(0).toUpperCase() + words[i].
slice(1);
}
return words.join(" ");
```

4.10 总有一种题，叫看起来都对（JSON）

第 241 道：解释一下 JSON 是如何进行跨域的？

解析：

Json 可以跨域，但是必须有 Jsoncallback 这个参数约定。

比如：

```
$.getJSON('http://www.qdjh.com/api/users/items/?username=desandro' + '&apikey=8b604e5d4841c2cd976241dd90d319d7' + '&tag=bestofisotope&callback=?')
```

对方必须有 `callback=?` 这个约定。如果没有这个约定，JSON 硬直跨域，难度太大，最后你会发现，费了九牛二虎之力还不如用服务器写个 `services` 去抓取，然后用 JSON 去请求自己这个 `services` 地址，因为服务器不涉及跨域安全性问题，JavaScript 是有这个安全性限制的。

第 242 道：请实现一个 `Json.stringify(Json)` 方法，能够对标准的 JSON 序列化。

解析：

JSON 对象有两个方法：`stringify()` 和 `parse()`。在最简单的情况下，这两个方法分别用于把 JavaScript 对象序列化为 JSON 字符串，以及把 JSON 字符串解析为原生 JavaScript，早期的 Json 解析器基本上就是使用 JavaScript 的 `eval()` 函数。由于 JSON 是 JavaScript 自己的语法，因此 `eval()` 函数可以解析、解释并返回 JavaScript 的对象和数组。ECMAScript 5 对解析 JSON 的行为进行了规范，定义了全局对象 `JSON`。

代码如下：

```
<html>
  <head>
    <title></title>
    <script type="text/javascript">
      function init(){
        var book={
          title:"JavaScript 高级程序设计 ",
          authors:["Nicholas C. Zakas"],
          edition:3,
          year:2011
        };
        var JsonBook=Json.stringify(book);
        var objectBook=Json.parse(JsonBook);
        var title=objectBook.title;}
    </script>
  </head>
  <body>
    <input type="button" onclick="init()" value=" 测试 " />
  </body>
</html>
```

默认情况下，`Json.stringify()` 输出的 JSON 字符串不包含任何空字符或缩进，因此保存在 `JsonBook` 中的字符串如下所示：

```
{ "title": "JavaScript 高级程序设计 ", "authors": ["Nicholas C. Zakas"], "edition": 3, "year": 2011 }
```

在序列化 JavaScript 对象时，所有函数及原型成员都会被有意忽略，不体现在结果中。此外，值为 `undefined` 的任何属性也都会被跳过。最后，结果中都是

值为有效 JSON 数据类型的实例属性。

注意，虽然 book 与 objectBook 具有相同的属性，但它们是两个独立的、没有任何关系的对象。如果传给 Json.parse() 的字符串不是有效的 JSON，该方法会显示错误。

第 243 道：说说 Jsonp 的实现方式。

解析：

最简单的 Jsonp 实现方式如下。

```
var Jsonp = document.createElement("script") ;
//Firefox:onload IE:onreadystatechange
Jsonp.onload = Jsonp.onreadystatechange = function(){
    //onreadystatechange, 仅 IE
    if (!this.readyState || this.readyState === "loaded" ||
        this.readyState === "complete") {
        alert($("#demo").html());
        Jsonp.onload = Jsonp.onreadystatechange = null// 清理
        防止 IE memory leaks
    }
}

Jsonp.type = "text/javascript";
Jsonp.src = "http://www.qdjh.com/2010/JS/jquery.js";
// 在 head 之后添加 JS 文件
document.getElementsByTagName("head")[0].appendChild(Jsonp);
```

简单解释：我们通过创建 script，指定它的 src 等属性，然后插入 head 执行。建议 onload、onreadystatechange 写在 src 赋值之前，防止载入 JavaScript 太快而没有给 onload、onreadystatechange 赋值（Image 对象在 IE 下具有此类现象）。

Jsonp 实例：

首先我们可以定义一个函数来执行 Jsonp 返回的数据，然后通过 Jsonp 的 src 传此函数给后台，进行处理，返回可执行的函数。例如下面代码：

```
function JsonpHandle(a){
    alert(a);
}

var Jsonp=document.createElement("script") ;
Jsonp.type="text/javascript";
Jsonp.src = http://www.qdjh.com/Jsonp.php?callback=JsonpHandle";
// 在 head 之后添加 JavaScript 文件
document.getElementsByTagName("head")[0].appendChild(Jsonp);
后台 Jsonp.php 的代码:
echo $_GET["callback"].("hello,world");
```

第 244 道：Jsonp 是解决跨域的唯一途径吗？可异步吗？实现机制是什么？
Jsonp 和 Json 有关系吗？

解析:

- (1) Jsonp 不是解决跨域的唯一途径。
- (2) 可以异步。
- (3) 实现机制如下。

Jsonp 的全称是 Jsonp with Padding, 是为了解决跨域请求资源而产生的解决方案。很多时候我们需要在客户端获取服务器数据进行操作, 一般我们会使用 Ajax+Web Service 做此事, 但是如果我们希望获取的数据和当前页面并不是一个域, 著名的同源策略(不同域的客户端脚本在没明确授权的情况下, 不能读写对方的资源)会因为安全原因拒绝请求, 也就是说, 我们不能向其他域直接发送请求以获取资源, 在 localhost 域上有一个 books.php, 里面包含脚本对 test.com 域的 books.php 发送 get 请求, 希望获取其 book 列表资源, 这就是一个跨域请求资源。

代码如下:

```
$.ajax({
    type: 'get',
    url: 'http://www.qdjhu.com/books.php'
});
```

页面会报一个这样的错误: XMLHttpRequest cannot load。而 http://www.qdjhu.com/books.php. Origin http://localhost is not allowed by Access-Control-Allow-Origin.Jsonp 是为了解决这个问题出现的。

(4) Jsonp 和 JSON 的关系。

Json (JavaScript Object Notation) 和 Jsonp (Json with Padding) 虽然只有一个字母的差别, 但其实它们根本不是一回事儿。JSON 是一种数据交换格式, 而 Jsonp 是一种依靠开发人员的聪明才智创造出来的一种非官方跨域数据交互协议。我们拿谍战片来打个比方, JSON 是地下党用来书写和交换情报的“暗号”, 而 Jsonp 则是把用暗号书写的情报传递给自己同志时使用的接头方式。

第 245 道: 解释 Jsonp 的原理, 以及它为什么不是真正的 Ajax。

解析:

1. Jsonp 原理

虽然有同源策略的限制, 但是并不是 HTML 上所有资源都必须是同一个域, 我们常见的页面为了节省流量或加载速度, 采用 Google 或微软的 jQuery CDN, 在页面上我们可以这样写以下代码, 此时就可以引用 jQuery 了。iframe、img、style、script 等元素的 src 属性可以直接向不同域请求资源, Jsonp 正是利用 script 标签实现跨域请求资源。

代码如下:

```
<script type="text/javascript">
src="http://www.qdjhu.com/Ajax/libs/jquery/1.7.1/"
```



```
jQuery.min.JS">
</script>
```

2. 原因

Jsonp 是需要动态创建 script 标签的，是回调函数。

Ajax 是页面无刷新请求数据操作。

4.11 最难回答的考题（事件委托）

第 246 道：假如一个父容器里有一万个子元素，要给它们全部绑定事件，如何绑定最好？

解析：

有这样一道 JavaScript 题目：在如下 DOM 结构中，如何高效地给 li 元素绑定 click 事件，从而当用户单击 li 元素时能够提示 li 中的内容？

```
<ul>
  <li>xxx</li>
  <li>xxx</li>
  <li>xxx</li>
  .....
  .....
</ul>
```

最初想法就是遍历每个 li 元素，循环给 li 绑定 onclick 事件，代码如下：

```
window.onload = function() {
  var ul = document.getElementById('ul');
  var lis = ul.getElementsByTagName('li');
  for (var i = lis.length-1; i >= 0; i--) {
    lis[i].onclick = function(e) {
      alert(this.innerHTML);
    }
  }
};
```

这种想法当然是最简单、最直观、也是正确的，但是存在一点问题，当 DOM 中的 li 元素特别多时，这样循环遍历的绑定操作势必会占用大量的资源，这时候我们可以使用事件的一些特性，将操作绑定到 ul 元素上面，当事件触发时，自动获取当前事件操作的对象，然后再完成操作。如下面代码：

```
window.onload = function() {
  var ul = document.getElementById('ul');

  ul.onclick = function(e) {
    e = window.event ? window.event : e;
```

```
var who = e.target ? e.target : e.srcElement;
alert(who.innerHTML);
};
};
```

这里我们用到了 Event 对象的 target 属性，该属性能够获取事件发生所在的元素。当然在 IE 下，该功能被 srcElement 代替。

第 247 道：描述一下事件冒泡，介绍事件委托（事件代理）的实现方法，以及它的原理和优点。

解析：

(1) 事件冒泡。在一个对象上触发某类事件（比如单击 onclick 事件），如果该对象定义了此事件的处理程序，那么此事件就会调用这个处理程序；如果没有定义此事件处理程序或者事件返回 true，那么这个事件会向这个对象的父级对象传播，从里到外，直至它被处理（父级对象所有同类事件都将被激活），或者它到达了对象层次的顶层，即 document 对象（有些浏览器是 window）。

(2) 事件委托的实现方法。通俗地讲，onclick、onmouseover、onmouseout 等就是事件。而委托，就是让别人来做，这个事件本来是加在某些元素上的，然而你却加到别人身上来做，以完成这个事件。也就是：利用冒泡的原理，把事件加到父级上，触发执行效果。

(3) 事件委托的原理。事件委托，其实从名字上就很好理解，就是自己的事件交给别人去做，即将事件委托给别人。

(4) 事件委托的优点。比如有很多 li 元素，我们想为每一个 li 元素注册一个单击事件，当然你可以将每个元素挨个进行注册，但是如果有 100 个或者更多个 li 元素的话，这绝对是一项海量的工作，如果我们利用冒泡原理，将事件委托给 li 元素的父级元素处理，那么无论有多少个 li 元素，都能够轻松搞定。

例如：

```
<!DOCTYPE html>
<html>
<head>
<meta charset=" utf-8">
<meta name="author" />
<title> 耶耶耶 </title>
<style type="text/css">
ul{
    list-style:none;
}
</style>
<script type="text/javascript">
window.onload=function(){
    var obox=document.getElementById("box");
```

```

var oshow=document.getElementById("show");
obox.onclick=function(ev){
    var ev=ev||window.event;
    var target=ev.target||ev.srcElement;
    oshow.innerHTML=target.innerHTML;
}
}
</script>
</head>
<body>
<div id="show"></div>
<ul id="box">
    <li>一</li>
    <li>二</li>
    <li>三</li>
    <li>四</li>
</ul>
</body>
</html>

```

第 248 道：事件委托是什么？

解析：

利用事件冒泡的原理，自己所触发的事件，由它的父元素代替执行。

4.12 高智商考题（事件流）

第 249 道：解释 JavaScript 事件冒泡机制。

解析：

事件：在浏览器客户端应用平台，基本上都是以事件驱动的，即某个事件发生，然后做出相应的动作。浏览器的事件表示的是某些事情发生的信号。

冒泡：例如，气泡从水底开始往上升，由深到浅，升到最上面。在上升的过程中，气泡会经过不同深度层次的水。这个气泡就相当于这里的事件，而水则相当于整个 DOM 树；事件从 DOM 树的底层一层一层往上传递，直至传递到 DOM 的根节点。

第 250 道：使用 JavaScript 语言，编写一个由小到大的冒泡排序算法。

解析：

```

<html>
<head>
<title> New Document </title>
<meta name="Generator" content="EditPlus">

```

```

<meta name="Author" content="">
<meta name="Keywords" content="">
<meta name="Description" content="">
</head>
<body>
<script language="javascript" type="text/javascript"> <!--
function bubbleSort(arr){
    // 外层循环，共要进行 arr.length 次的求最大值的操作
    for(var i=0;i<arr.length;i++){
        // 内层循环，找到第 i 大的元素，并将其和第 i 个元素交换
        for(var j=i;j<arr.length;j++){
            if(arr[i]<arr[j]){ // 交换两个元素的位置
                var temp=arr[i];
                arr[i]=arr[j];
                arr[j]=temp;
            }
        }
    }
}

var arr=[32,55,78,43,78,10,45,20,9,89],
    bubbleSort(arr);
// 输出: 89,78,78,55,45,43,32,20,10,9
for(var i=0;i<arr.length;i++){
    document.write(arr[i]+",");
}
//-->
</script>
</body>
</html>

```

第 251 道：分别写出普通与绑定的阻止默认事件。

解析：

普通：

```
return false;
```

绑定：

```

preventDefault();
if(oEvent.preventDefault) {
    oEvent.preventDefault();
}
return false;

```

第 252 道：事件处理程序及事件对象的跨浏览器实现。

解析：

```
varEventUtil = (function() {
```

```

var EventUtil,    // 事件处理对象
eventList = [],  // 事件列表
// 浏览器兼容的鼠标滚动事件 type
mouseWheelType = typeof window.opera !== "object" && !/
(Khtml|AppleWebKit|Konqueror)/.test(navigator.userAgent) &&
/Gecko/.test(navigator.userAgent) ? "DOMMouseScroll":
"mousewheel", // 鼠标滚动事件 type 正则
mouseWheelRegExp = /^mousewheel|DOMMouseScroll$/;
// 事件详细信息构造器
function EventDetail(element, type, handler, proxy) {
    this.element = element;
    this.type = type;
    this.handler = handler;
    this.proxy = proxy;
}
// 将事件对象处理成 DOM 标准的事件对象
function eventObjHandler(event, element) {
    event = event || window.event;
    if(!event.target) {
        event.target = event.srcElement;
    }
    if(event.cancelable === undefined) {
        event.cancelable = true;
    }
    if(!event.preventDefault) {
        event.preventDefault = function() {
            event.returnValue = false;
        };
    }
    if(event.bubbles === undefined) {
        event.bubbles = true;
    }
    if(!event.stopPropagation) {
        event.stopPropagation = function() {
            event.cancelBubble = true;
        };
    }
    if(!event.currentTarget) {
        event.currentTarget = element;
    }
    if(!event.eventPhase) {
        event.eventPhase = event.target === event.
currentTarget ? 2: 3;

```

```

    }
    if(mouseWheelRegExp.test(event.type) && !event.
wheelDelta) {
        event.wheelDelta = -event.detail * 40;
    }
    return event;
}
// 创建 handler 的代理函数，用于修正 this 引用
function createProxyFunc(ele, type, handler) {
    var proxy = function(event) {
        handler.call(ele, eventObjHandler(event, ele));
    };
    eventList.push(newEventDetail(ele, type, handler,
proxy));
    return proxy;
}
// 从 eventList 中找出真正绑定到事件中的处理函数，并删除相应的
EventDetail
function findTheTrueEventHandler(ele, type, handler) {
    var i, eventDetail;
    for(i = eventList.length - 1; i >= 0; i--) {
        eventDetail = eventList[i];
        if(eventDetail.element === ele && eventDetail.
type === type && eventDetail.handler === handler) {
            break;
        }
    }
    if(i > -1) {
        handler = eventList[i].proxy; eventList.
splice(i, 1);
    }
    return handler;
}
// 修正鼠标滚轮事件名
function repairMouseWheel(type) {
    return mouseWheelRegExp.test(type) ? mouseWheelType :
type;
}
EventUtil = {
    // 添加事件绑定
    add: function(ele, type, handler) {

```

```

        type = repairMouseWheel(type); // 修正 type
        if(ele.addEventListener) { // 标准
            ele.addEventListener(type,
                type === "DOMMouseScroll"?
                createProxyFunc(ele, type, handler) : handler, false);
        } else if(ele.attachEvent) { // IE
            ele.attachEvent("on"+ type,
                createProxyFunc(ele, type, handler));
        } else { // DOM1
            ele["on"+ type] = function(event) {
                handler.call(ele, eventObjHandler(event));
            };
        }
    },
    // 移除事件绑定
    remove: function(ele, type, handler) {
        type = repairMouseWheel(type); // 修正 type
        if(ele.removeEventListener) { // 标准
            ele.removeEventListener(type,
                type === "DOMMouseScroll"?
                findTheTrueEventHandler(ele, type, handler) :
                handler, false);
        } else if(ele.detachEvent) { // IE
            ele.detachEvent("on"+ type,
                findTheTrueEventHandler(ele, type, handler));
        } else { // DOM1
            ele["on"+ type] = null;
        }
    }
};

// 绑定页面关闭事件，当页面关闭时移除所有事件
EventUtil.add(window, "unload", function() {
    var i, eventDetail;
    for(i = eventList.length - 1; i >= 0; i--) {
        eventDetail = eventList[i];
        if(eventDetail.type === "DOMMouseScroll") {
            eventDetail.element.removeEventListener
                (eventDetail.type, eventDetail.proxy, false);
        } else {
            eventDetail.element.detachEvent("on"+
                eventDetail.type, eventDetail.proxy, false);
        }
    }
});

```

```

    }
    eventList[i] = null;
  }
  eventList = [];
});
return EventUtil;
})();

```

第 253 道：event.preventDefault 的作用是什么？与 event.stopPropagation 有什么区别？请举例说明。

解析：

event.preventDefault() 和 event.stopPropagation() 不是 jQuery 的方法，而是 js 本身自带的。

event.preventDefault() 用法：介绍该方法将通知 Web 浏览器不要执行与事件关联的默认动作（如果存在这样的动作）。例如，如果 type 属性是“submit”，在事件传播的任意阶段可以调用任意的事件句柄，通过调用该方法，可以阻止提交表单。注意，如果 Event 对象的 cancelable 属性是 false，那么就没有默认动作，或者不能阻止默认动作。无论哪种情况，调用该方法都没有作用。

event.stopPropagation() 用法：介绍该方法将停止事件的传播，阻止它被分派到其他 document 节点。在事件传播的任何阶段都可以调用它。注意，虽然该方法不能阻止同一个 document 节点上的其他事件句柄被调用，但是它可以阻止把事件分派到其他节点。

第 254 道：如何阻止冒泡事件和默认事件？

解析：

阻止冒泡事件发生需要调用以下函数：

```

function stopBubble(e){
    if(e&&e.stopPropagation){//非IE
        e.stopPropagation();
    }else{//IE
        window.event.cancelBubble=true;
    }
}

```

阻止默认事件发生需要调用以下函数：

```

function stopDefault( e ) {
    // 阻止默认浏览器动作 (W3C)
    if ( e && e.preventDefault )
        e.preventDefault();
    //IE 中阻止浏览器默认动作的方式
    else
        window.event.returnValue = false;
}

```



```
        return false;
    }
}
```

4.13 前端压轴考题（错误处理与调试）

第 255 道：JavaScript 程序中异常捕获的方法有哪些？

解析：

使用 Try...catch...来异常捕获（主要适用于 IE5 以上内核的浏览器，也是最常用的异常捕获方式）。

使用 onerror 事件异常捕获，这种捕获方式是比较古老的一种方式，目前一些主流的浏览器暂不支持此种捕获方式。

4.14 你值得拥有的考题（Cookie）

第 256 道：写一个设置 Cookie 值的封装函数。

解析：

- (1) encodeURIComponent() 转换字符编码为统一编码（涉及中文编码）。
- (2) toUTCString() 将时间转换为字符串，Cookie 只能接收字符串形式。

```
function setCookie(key, value, t) {
    var oDate = new Date();
    oDate.setDate(oDate.getDate() + t);
    document.Cookie = key + '=' + encodeURIComponent(value)
    + ';expires=' + oDate.toUTCString();
}
```

第 257 道：列举 3 ~ 6 个 Cookie 的属性。

解析：

Cookie 的属性有 Comment、Domain、Max-Age、Path、Secure、Version。

第 258 道：简述一下 Cookie 的操作，以及 Cookie 的属性都有哪些？

解析：

Cookie 是浏览器提供了一种机制，它将 document 对象的 Cookie 属性提供给 JavaScript。可以由 JavaScript 对其进行控制，而并不是 JavaScript 本身的性质。Cookie 是存于用户硬盘的一个文件，这个文件通常对应于一个域名，当浏览器再次访问这个域名时，便使这个 Cookie 可用。因此，Cookie 可以跨越一个域名下的多个网页，但不能跨越多个域名使用。可用在保存用户的登录状态、跟踪用户行为、定制页面、创建购物车。

```
$.Cookie("CookieName","CookieValue",{expires:7,path:"/",domain:"qdjhu.com",secure:false,raw:false});
```

expires: 有效时间; path: 设置能够读取 Cookie 的顶级目录; domain: 创建 Cookie 所在网页所拥有的域名; secure: 默认是 false, 如果为 true, Cookie 的传输协议为 https; raw: 默认为 false, 读取和写入时自动进行编码和解码 (使用 encodeURIComponent 编码, 使用 decodeURIComponent 解码), 要关闭这个功能, 需设置为 true。

综合提升

第 259 道: 异步加载 JavaScript 的方式有哪些? 请分别举例。

解析:

1. \$.getScript(URL, callback)

这个方法提供了异步加载 script 资源的方式, 对于一些 Web 网页内容比较多, 需要按需加载的情况, 提供了很大的帮助, jQuery1.2 之后的这个方法可以跨域访问, 它通过动态创建 script, 在加载成功后删除 script 节点。使用方法: \$.getScript("/JS/user.JS");

2. \$.getJSON()

该方法提供了访问同一个域中的 JSON 数据。

```
$("#AjaxLoadJson").click(function(){
    $.getJSON("JS/user.Json", function(data) {
        $("#divTip").empty(); // 先清空标记中的内容
        var strhtml = ""; // 初始化保存内容变量
        $.each(data, function(InfoIndex, Info) { // 遍历获取的数据
            strhtml += "姓名: " + Info["name"] + "<br>";
            strhtml += "性别: " + Info["sex"] + "<br>";
        })
        $("#divTip").html(strhtml); // 显示处理后的数据
    })
});
```

对应的 user.Json:

```
[
  {
    "name": "a",
    "sex": "男",
    "email": "a@163.com"
  },
  {
    "name": "b",
```

```

        "sex": "女",
        "email": "b@163.com"
    }
}

```

URL 表示请求的地址, data 表示请求的参数, 可选参数 CALLBACK 在回调函数中执行操作。

3. \$("#div").load(URL selector)

该方法提供了异步获取 HTML 数据的方式, 这个方法也不能跨域访问, 在 URL 后面可以指定异步请求的网页的哪些部分被加载到该 div 中, 举个例子:

```

$("#AjaxLoadhtml").click(function() { // 按钮单击事件
    $("#Ajax").load("index.Jsonp h3"); //load() 方法加载数据
})

```

index.JSP 代码

```

<html>
<head>
</head>
<body>
<h2>Hello World!</h2>
<h3>你好</h3>
</body>
</html>

```

第 260 道: 请简述同步和异步的区别。

解析:

同步是指发送方发出数据, 等接收方发回响应以后才发下一个数据包的通信方式。

异步是指发送方发出数据后, 不等接收方发回响应, 接着发送下一个数据包的通信方式。

第 261 道: 请写一段代码, 判断当前脚本运行在浏览器中还是 node 环境中。

解析:

判断 Global 对象是否为 window, 如果不为 window, 那么当前脚本没有运行在浏览器中。

第 262 道: location.search() 是什么?

解析:

设置或获取网页地址跟在问号后面的部分。

当以 get 方式在 URL 中传递了请求参数时, 可以利用 location 的 search 属性提取参数的值。

第 263 道: 模拟一个 HashTable 类, 包含 add、remove、contains、length 方法。

解析:

```
function Hashtable()
```

```
{
  this.container = new Object();
  /**/** put element */
  this.put = function (key, value)
  {
    if (typeof (key) == "undefined")
    {
      return false;
    }
    if (this.contains(key))
    {
      return false;
    }
    this.container[key] = typeof (value) ==
    "undefined" ?null : value;
    return true;
  };
  /**/** remove element */
  this.remove = function (key)
  {
    delete this.container[key];
  };
  /**/** get size */
  this.size = function ()
  {
    var size = 0;
    for (var attr in this.container)
    {
      size++;
    }
    return size;
  };
  /**/** get value by key */
  this.get = function (key)
  {
    return this.container[key];
  };
  /**/** contains a key */
  this.contains = function (key)
  {
    return typeof (this.container[key]) != "undefined";
  };
}
```

```

    /**/** clear all entrys */
    this.clear = function ()
    {
        for (var attr in this.container)
        {
            delete this.container[attr];
        }
    };
    /**/** hashTable 2 string */
    this.toString = function()
    {
        var str = "";
        for (var attr in this.container)
        {
            str += "," + attr + "=" + this.container[attr];
        }
        if(str.length>0)
        {
            str = str.substr(1, str.length);
        }
        return "{" + str + "}";
    };
}

```

第 264 道：JavaScript 如何实现面向对象和继承机制？

解析：

面向对象：

```

function MyObject(){
    this.name = "myObject";
    this.type = "class";
    this.methodA = function(){
        alert(this.name);
    }
    this.methodB = function(){
        return this.type;
    }
}

var myObject = new MyObject();
myObject.methodA();
var type = myObject.methodB();
alert(type);

```

继承机制：

(1) 构造继承适合单个 class，优点是继承关系明确。

(2) 原型链方式，适合无参数继承。

(3) 混合方式，根据情况而定。

第 265 道：请看下面这段代码：

```
var a=document.getElementsByTagName("a");
for(var i=0;i<a.length;i++){
    a[i].onclick=function(){
        alert(i);
    }
}
```

本程序要完成的功能是：单击页面中的超链接时，弹出该链接的编号。请问该程序执行后，单击页面中的超链接时，会弹出什么值？然后修改这段代码来达到预期效果。

解析：

会弹出 a 元素的个数

```
var a=document.getElementsByTagName("a");
for(var i=0;i<a.length;i++){
    a[i].onclick=(function(i){
        alert(i);
    })(i);
}
```

第 266 道：编写一个类，类中定义如下属性和方法：共有属性和共有方法、共有静态属性和共有静态方法、私有属性和私有方法、特权属性和特权方法、静态属性和静态方法。然后，在代码中通过注释的形式注明上述每个属性和方法。

解析：

```
function User(){
    // 公有属性:
    // 特权属性:
    this.name='byronvis';
    // 特权方法:
    // 公有方法:
    this.sayName=function(){
        alert(this.name);
        alert(this.school);
        alert(age);// 变量声明会自动提前
        alert(this.sex);
    };
    // 私有属性:
    var age=22;
    // 私有方法:
    function sayAge(){
```

```

        alert(age);
    }
    sayAge();
}

// 共有属性：共享内存
// 共有静态属性
User.prototype.school='zky';
// 共有方法：可访问公有属性
// 共有静态方法
User.prototype.saySchool=function(){
    alert(this.school);
    alert(this.name);
    alert(this.sex);
    alert(age);
};
var obj=new User();
// 静态属性：就是动态添加的实例属性
obj.sex='man';
// 静态方法：就是动态添加的实例方法
obj.saySex=function(){
    alert(this.sex);
    alert(this.name);
    alert(this.school);
    alert(age);
};

```

第 267 道：JavaScript 里面的基础对象和基础数据类型有哪些？

解析：

基础对象：

1. 函数构造法

```
function funcName(){};
```

2. 对象生成法

```
var obj={};
obj.x=1;
obj.y=2;
```

3. 对象直接生成法

```
var obj={x:1,y:2}
```

基础数据类型：JavaScript 中有 5 种简单数据类型（也称为基本数据类型）：Undefined、Null、Boolean、Number 和 String。还有一种复杂数据类型——Object，Object 本质上是由一组无序的名值对组成的。

鉴于 JavaScript 是松散类型的，因此需要有一种手段来检测给定变量的数据

类型，而 `typeof` 就是负责提供这方面信息的操作符。对一个值使用 `typeof` 操作符可能返回下列某个字符串：

- "undefined"——如果这个值未定义；
- "boolean"——如果这个值是布尔值；
- "string"——如果这个值是字符串；
- "number"——如果这个值是数值；
- "object"——如果这个值是对象或 `null`；
- "function"——如果这个值是函数。

第 268 道：怎样将一个过大的模块分开放在不同的 JavaScript 文件中？

解析：

最早的时候，所有 JavaScript 代码都写在一个文件里面，只要加载这一个文件就够了。后来，代码越来越多，一个文件不够了，必须分成多个文件，依次加载。下面的网页代码，相信很多人都见过。

代码如下：

```
<script src="1.JS"></script>
<script src="2.JS"></script>
<script src="3.JS"></script>
```

这段代码依次加载多个 JavaScript 文件。

这样的写法有很大的缺点。首先，加载时，浏览器会停止网页渲染，加载文件越多，网页失去响应的时间就会越长；其次，由于 JavaScript 文件之间存在依赖关系，因此必须严格保证加载顺序（比如上例的 1.JS 要在 2.JS 的前面），依赖性最大的模块一定要放到最后加载，当依赖关系很复杂时，代码的编写和维护都会变得困难。

`require.JS` 的诞生，就是为了解决以下这两个问题：

- （1）实现 JavaScript 文件的异步加载，避免网页失去响应；
- （2）管理模块之间的依赖性，便于代码的编写和维护。

第 269 道：请解释一下 JavaScript 的本地对象，内置对象和数组对象。

解析：

1. 本地对象

ECMA-262 把本地对象（native object）定义为“独立于宿主环境的 ECMAScript 实现提供的对象”。

本地对象包含以下内容：

`Object`、`Function`、`Array`、`String`、`Boolean`、`Number`、`Date`、`RegExp`、`Error`、`EvalError`、`RangeError`、`ReferenceError`、`SyntaxError`、`TypeError`、`URIError`。

由此可以看出，本地对象就是 ECMA-262 定义的类（引用类型）。

2. 内置对象

ECMA-262 把内置对象 (built-in object) 定义为“由 ECMAScript 实现提供的、独立于宿主环境的所有对象，在 ECMAScript 程序开始执行时出现”。这意味着开发者不必明确实例化内置对象，它已被实例化了。

同样是“独立于宿主环境”。根据定义我们似乎很难分清“内置对象”与“本地对象”的区别。而 ECMA-262 只定义了两个内置对象，即 Global 和 Math 根据定义，每个内置对象都是本地对象，所以它们也是本地对象。

如此就可以理解了，内置对象是本地对象的一种，而其包含的两种对象中，Math 对象我们经常用到，可 Global 对象是什么呢？

Global 对象是 ECMAScript 中最特别的对象，因为实际上它根本不存在，但大家要清楚，在 ECMAScript 中，不存在独立的函数，所有函数都必须是某个对象的方法。类似于 isNaN()、parseInt() 和 parseFloat() 方法等，看起来都是函数，而实际上，它们都是 Global 对象的方法。而且 Global 对象的方法还不止这些。

3. 数组对象

由 ECMAScript 实现的宿主环境提供的对象，可以理解为浏览器提供的对象。所有的 BOM 和 DOM 都是数组对象。

第 270 道：解释一下 JavaScript 的同源策略。

解析：

同源策略限制了一个源 (origin) 中加载文本或脚本与来自其他源中资源的交互方式。

第 271 道：给你一个 100×100 的照片，每单击一下照片绕中心点旋转 30° 。

解析：

- (1) 将所有图片放置在合成中心。
- (2) 用 360 除以图片数量，得到每个图片旋转的角度。比如你现在有 12 张图片，每一张图片旋转角度为 30° (第一张不动，第二张旋转 30° ，第二张旋转 60° ，第三张旋转 90° ，以此类推)。
- (3) 旋转完了所有图片之后，选择所有图片层，按下 A 键调出定位点关键帧，拖动 Y 轴数值让所有的图片散开，形成一个环。
- (4) 通过定位点的调节得到你想要的大小之后，在合成中间添加一个空白对象层。
- (5) 选择所有的图片层，将父级下面的橡皮带全部扯到空白对象上面。
- (6) 单独选中空白对象层，R 键打开旋转属性，旋转空白对象的同时所有的图片都跟着旋转了。
- (7) 把空白对象移动到你给的那张示例图的圆圈中心位置 (图片会跟着走)。
- (8) 设置空白对象的旋转关键帧即可。

第 272 道：你能解释一下 JavaScript 中的继承是如何工作的吗？

解析：

- (1) 在子构造函数中执行父构造函数，并用 `call/apply` 改变 `this`。
- (2) 克隆父构造函数原型上的方法。

第 273 道：简单阐述设计模式中工厂方法模式、单列模式和观察者模式。

解析：

1. 工厂方法模式

一个抽象产品类，可以派生出多个具体产品类。

一个抽象工厂类，可以派生出多个具体工厂类。

每个具体工厂类只能创建一个具体产品类的实例。

2. 单列模式保证类只有一个实例

3. 观察者模式

举例阐述：游戏情节，一个小男孩，丢到众多鬼子附近，爆炸啦，根据炸弹的威力计算爆炸后鬼子的血量，假定有些鬼子有防具，有些鬼子没有防具。

分析：这种情况，使用观察者模式是比较理想的，因为观察者模式就是处理对象间一对多的依赖关系的，当一个对象发生变化，其他依赖他的对象都要得到通知并更新。

定义：在观察者模式中，上述小男孩被称为主题，而小鬼子们就被称为观察者。

274 道：JavaScript 实现数组升序排序。

解析：

```
var arr=[2,3,5,1,15,8,12,11,7];
function des(a,b){
    return a-b;
}
console.log(arr.sort(des));
```

第 275 道：常见的浏览器内核有哪些？

解析：

- (1) IE 浏览器的内核：Trident。
- (2) Mozilla 的 Gecko。
- (3) Chrome 的 Blink（WebKit 的分支）。
- (4) Opera 的内核原为 Presto，现为 Blink。

第 276 道：请简单说明 JavaScript 实现拖曳的原理。

解析：

(1) 拖曳的基本原理：当 `mousedown` 时，记下鼠标单击位置离拖曳容器左边沿的距离和上边沿的距离，即 `tmpX/tmpY`；当 `mousemove` 时，通过定位拖曳容器的 `style.left/style.top`，使拖曳容器进行移动，定位到哪里则由刚刚的 `tmpX/`

tmpY 和当前鼠标所在的位置计算得出；当 mouseup 时，结束移动。

(2) “var dragObj = this;” 这句是为了在 mousedown/mouseup/mousemove 事件里对 Drag 对象的相关变量进行引用。因为在 mousedown 里的 this 是 titleBar, 而 mouseup/mousemove 里的 this 是 document。

(3) 当拖曳速度太快导致鼠标移出拖曳容器，而拖曳容器位置未变时，用 document.mousemove 代替 titleBar.mousemove 即可。

(4) 设置拖曳容器可拖曳的范围，若未设置，则默认为当前窗口可视范围。Note：在设置范围时使用 Math.max/min 来处理，而不是用 if 语句判断，用后者的话会导致快速拖曳时未达到容许范围边沿即停止的状况。

(5) 在拖曳过程中，可设置是否保留原来拖曳容器，当拖曳结束时，隐藏原来容器，默认不保留。

(6) 当拖曳时，可设置拖曳的容器是否透明及透明度是多少，默认为不透明。但若拖曳过程中设置保留原来拖曳容器，即 keepOrigin: true, 则设置透明度为 50%。

(7) 单击鼠标右键、鼠标中键等不能拖动，仅单击鼠标左键可以拖动。Note：IE 鼠标左键为 event.Button=1；Firefox 鼠标左键为 event.Button=0。

(8) 解决单击图片无法拖曳的问题：非常“杯具”的是 IE 通过 ev.returnValue = false 来防止图片的事件，注意是放在 document.onmousemove 中，而 FireFox 通过 ev.preventDefault();ev.stopPropagation(); 来防止图片的事件，但是却放在 titleBar 的 mousedown 事件中。

(9) 有一种情况，当浏览器窗口不是最大化时，你希望鼠标在浏览器外移动时浏览器里的拖曳容器仍然移动，这时就要使用鼠标事件捕获，IE 中相应的是 dragdiv.setCapture(); 与 dragdiv.releaseCapture(); 而 Firefor 中相应的是 window.captureEvents(Event.mousemove); 与 window.releaseEvents(dragdiv.mousemove)。

(10) 确保当每次拖曳时，拖曳容器中的 zIndex 都不会被其他块元素覆盖。

第2篇

HTML5+CSS3 我独行 —— 驾骏马， 拉长弓

HTML5+CSS3 不仅是两项新的 Web 技术标准，而且代表了下一代 HTML 和 CSS 技术。各大主流浏览器厂家已经积极更新自己的产品，以便好地支持 HTML5。HTML5 与 CSS3 相辅相成，使互联网进入了一个崭新的时代。

第 5 章



博学多才，雄韬伟略 [HTML5+CSS3 初级面试题]

能文能武之人方为可怕之人，HTML5+CSS3 将会祝你大展雄风。天下第一神器“Photoshop”刀光剑影，未闻其人先见其影；葵花点穴手（position）瞬间让你驻足而观。

5.1 做了这些，不再是菜鸟（关于 HTML5）

第 277 道：您有没有关注过 HTML5 和 CSS3？如有请简单说一下您对它们的了解情况。

解析：

HTML5 和 CSS3 不仅是两项新的 Web 技术标准，而且代表了下一代 HTML 和 CSS 技术。其未来的发展前景已经可以预见，那就是 HTML5 必将被越来越多的 Web 开发人员所使用。各大主流浏览器厂家已经积极更新自己的产品，以更好地支持 HTML5。它的优势主要有以下几点：

- （1）更多的描述性标签：HTML5 引入非常多的描述性标签。
- （2）良好的多媒体支持：对于先前以插件的方式播放音频、视频带来的麻烦，HTML5 有了解决方案，audio 标签和 video 标签能够方便地实现应变。
- （3）更强大的 Web 应用：HTML5 提供了令人称奇的功能，在某些情况下，你甚至可以完全放弃使用第三方技术。
- （4）跨文档消息通信：Web 浏览器会组织不同域间的脚本交互或影响，但是对于可信任的脚本或许就是麻烦。HTML5 引入了一套安全且易于实现的应对方案。
- （5）Web Sockets：HTML5 提供了对 Web Sockets 的支持。
- （6）客户端存储：HTML5 的 Web Storage 和 Web SQL Database API，可以在浏览器中构建 Web 应用的客户端持久化数据。
- （7）更加精美的界面：HTML5+CSS3 组合渲染出来的界面效果更加精美。
- （8）更强大的表单：HTML5 提供了功能更加强大的表单界面控件，使用非常方便。
- （9）提升可访问性：内容更加清晰，使用户的操作更加简单方便，提升用

户体验。

(10) 先进的选择器：CSS3 选择器可以方便地识别出表格的奇偶行、复选框等，代码标记更少。

(11) 视觉效果：具有精美的界面，有阴影、渐变、圆角、旋转等视觉效果。

CSS3 是 CSS 技术的升级版本，CSS3 语言开发是朝着模块化发展的。作为一个模块，以前的规范太庞大，而且比较复杂，所以，把它分解为一些小的模块，使更多新的模块被加进来。这些模块包括：盒子模型、列表模块、超链接方式、语言模块、背景和边框、文字特效、多栏布局等。

第 278 道：如果把 HTML5 看成一个开放平台，那它的构建模块有哪些？

解析：

如果把 HTML5 看成一个开放平台，它的构建模块至少应该包括以下几个：`<nav>`、`<header>`、`<section>`、`<footer>`。

`<nav>` 标签用来将具有导航性质的链接划分在一起，使代码结构在语义化方面更加准确。

`<header>` 标签用来定义文档的页眉（介绍信息）。

`<section>` 标签用来描述文档的结构。

`<footer>` 标签用来定义 section 或 document 的页脚。在典型情况下，该元素会包含创作者的姓名、文档的创作日期及联系信息。

5.2 真本事，更自信（HTML5 语法）

第 279 道：HTML 标记 `<pre>...</pre>` 表示 _____

解析：

`<pre>` 是预格式化标记，被包围在 `pre` 元素中的文本通常会保留空格和换行符。而文本也会呈现为等宽字体。

例如：

```
<html>
<head>
<title>pre</title>
</head>
<body>
  <pre>
    你好！
  你好！
  </pre>
</body>
</html>
```

显示的结果就是：

你好！

你好！

第 280 道：HTML5 不支持下面哪个元素（ ）

A. <p> B. <ins> C. <menu> D.

答案：D

解析：

 标签规定文本的字体外观、字体尺寸和字体颜色。在 HTML5 中不使用该元素而使用 CSS 向元素添加样式。

第 281 道：HTML5 不支持下面哪个元素（ ）

A. <cite> B. <acronym> C. <abbr> D. <hase>

答案：B

解析：

<acronym> 标签定义首字母缩写，比如“NASA”，通过对首字母缩写进行标记，您就能够为浏览器、拼写检查程序、翻译系统，以及搜索引擎分度器提供有用的信息。在 HTML5 中不支持 <acronym>，而使用 <abbr> 代替。

第 282 道：在 HTML5 中，哪个元素用于组合标题元素（ ）

A. <group> B. <header> C. <headings> D. <hgroup>

答案：D

解析：

<hgroup> 标签用于对网页或区段（section）的标题进行组合。<group> 将一组元素声明归在一起，以便将它们作为一个组合并到复杂类型定义中。

第 283 道：哪个 HTML5 元素用于显示已知范围内的标量测量（ ）

A. <gauge> B. <range> C. <measure> D. <meter>

答案：D

解析：

<meter> 标签定义度量衡。仅用于已知最大和最小值的度量。

第 284 道：描述一下 section、article、nav、footer 这几个 HTML5 的标签定义。

解析：

<section> 标签定义文档中的节（section、区段）。比如章节、页眉、页脚或文档中的其他部分。

<article> 标签定义外部的内容。外部内容可以来自一个外部的新闻提供者的一篇新的文章，或者来自 blog 的文本，或者来自论坛的文本，抑或来自其他外部源内容。

<nav> 标签定义导航链接的部分。

<footer> 标签定义 section 或 document 的页脚。

5.3 这些题总能温暖你（HTML5+CSS3 新增属性）

第 285 道：CSS3 有哪些新内容，请至少说出 5 个。

解析：

(1) CSS3 圆角表格，对应属性：border-radius;。

(2) 以往对网页上的文字加特效只能用 filter 属性，但是在 CSS3 中专门制订了一个加文字特效的属性，而且不止加阴影这种效果。对应属性：font-effect;。

(3) 丰富了对链接下画线的样式，以往的下画线都是直线，这次可不一样了，有波浪线、点线、虚线等，更可对下画线的颜色和位置进行任意改变（还有对应顶线和中横线的样式，效果与下画线类似）。对应属性：text-underline-style;text-underline-color;text-underline-mode;text-underline-position;。

(4) 在文字下画几个点或打个圈以示重点，CSS3 也开始加入了这项功能，这应该在某些特定网页上很有用。对应属性：font-emphasize-style; font-emphasize-position;。

(5) Font-face 可以用来加载字体样式，而且它还能够加载服务器端的字体文件，显示客户端没有安装的字体。

第 286 道：HTML5 有哪些新内容，请至少写出 5 个。

解析：

HTML5 增添了不少新标签和新属性，能使页面更优化，代码更简洁。例如：

(1) HTML5 已经确定引入 canvas 标签，通过 canvas，用户可以动态地生成各种图形图像、图表及动画。canvas 标签还能够配合 JavaScript 利用键盘来控制图形图像。

(2) 在 HTML5 中包含 Web Forms 2.0，用来描绘如何进行页面表格操作。其中最大的特点就是“表格确认”。当前，开发者通常使用 JavaScript（客户端）和 PHP（服务端）代码来确认输入的内容。

(3) HTML5 中为新元素和现有的元素提供更多的 API，旨在改进页面程序开发和增加 HTML4 所缺乏的特性。比如，一个视频和音频方面的 API 将与 <audio> 和 <video> 元素一起使用，它将提供视频和音频回放功能，而无需依赖第三方程序，比如 Flash。

(4) HTML5 中的 User Interaction 用来描述页面内容交互工作的新方式。它的 content editable 属性可以让开发者决定，页面哪部分内容允许进行用户更改，这对于 wiki 类的网站更为有用。

(5) HTML 的主要任务是描述页面的架构，例如在 <p>...</p> 元素之间的文本内容，HTML 将告诉浏览器这些文本是一个段落。

第 287 道：HTML5 新增的语义化标签有哪些？

解析:

HTML5 新增的语义化标签有很多, 比如说:

- `<article>` 标签定义外部的内容, 可以是一篇新的文章。
- `<aside>` 标签定义 `article` 以外的内容, `aside` 的内容可用作文章的侧栏。
- `<figcaption>` 标签定义 `figure` 元素的标题。
- `<figure>` 标签用于对元素进行组合, 使用 `figcaption` 元素为元素组添加标题。
- `<footer>` 标签定义 `section` 或文档的页脚。
- `<header>` 标签定义文档的页眉。
- `<hgroup>` 标签用于对 `section` 或网页的标题进行组合, 使用 `figcaption` 元素为元素组添加标题。
- `<nav>` 标签定义导航链接的部分。
- `<section>` 标签定义文档中的节 (`section`、区段)。比如章节、页眉、页脚或文档中的其他部分。

`<time>` 标签定义日期或时间。

第 288 道: HTML5 新增的属性有哪些?

解析:

HTML5 新增的属性有 `auto complete`、`min`、`max`、`multiple`、`pattern` 和 `step`。还有 `list` 属性与 `datalist` 元素配合使用; `datalist` 元素与 `autocomplete` 属性配合使用。`multiple` 属性允许一次上传多个文件; `pattern` 属性用于验证输入字段的模式, 其实就是正则表达式。`step` 属性规定输入字段的合法数字间隔 (假如 `step="3"`, 则合法数字应该是 -3、0、3、6, 以此类推), `step` 属性可以与 `max` 及 `min` 属性配合使用, 以创建合法值的范围。

为 `input`、`button` 元素增加 `formaction`、`formenctype`、`formmethod`、`formnovalidate` 和 `formtarget` 属性。用户重载 `form` 元素的 `action`、`enctype`、`method`、`novalidate` 和 `target` 属性。为 `fieldset` 元素增加 `disabled` 属性, 可以把它的子元素设为 `disabled` 状态。

为 `input`、`button`、`form` 增加 `novalidate` 属性, 可以取消提交时进行的有关检查, 表单可以被无条件地提交。

5.4 最实用的题 (HTML5 与 XML)

第 289 道: 在 HTML5 中如何使用 XML?

解析:

HTML 和 XML 有着显著的差别, 尤其是在解析需求方面, 并且无法使用针

对一方设计的工具去处理另一方的问题。但是，由于 HTML5 是根据 DOM 定义的，所以在大多数情况下，可使用 HTML 或 XHTML 序列化来表示同一文档。

第 290 道：HTML5 的页面中可以使用 XHTML 的语法吗？

解析：

可以，HTML5 向下兼容所有存在的 HTML 语法。

通过不同的头部声明 DTD 来设置怪异模式和标准模式。标准模式下，HTML4.0 提供了三种 DOCTYPE (DOCument TYPE, 文档类型)，XHTML1.0 提供了三种类型，它们是 transitional (过渡型)、strict (严格型)、frameset (框架型)。XHTML 可以理解为 HTML+XML，就是用 XML 的语法来规范 HTML。其实，早期的 HTML 语言是比较散漫的，像 `<p>1234532` 是可以被浏览器正常解析的，后来引入的 XHTML 和相应的 DTD，将 HTML 语言规范化，结构化。

第 291 道：找几条 XHTML 规范的内容。

解析：

- (1) 所有的标记都必须要有个相应的结束标记。
- (2) 所有标签的元素和属性的名字都必须使用小写。
- (3) 所有的 XML 标记都必须合理嵌套。
- (4) 所有的属性必须用引号 ""。
- (5) 所有的 `<` 和 `&` 特殊符号都要用编码表示。

5.5 KO 这些题，前端岗位不是梦 (HTML5 结构)

第 292 道：请用 HTML5 标签写一个符合语义化的页面，页面中有导航栏、页眉、页脚、文字内容以及图片内容。

解析：

```
<!DOCTYPE html>
<html>
<head>
  <Title>Page title</title>
</head>
<body>
  <header>
    <h1>Page title</h1>
  </header>
  <nav>
    <!--Navigation-->
  </nav>
  <section id="intro">
    <!--Introduction-->
```

```

</section>
<section>
    <!--Main content area-->
</section>
<aside>
    <!--Sidebar-->
</aside>
<footer>
    <!--Footer-->
</footer>
</body>
</html>

```

第 293 道：在 HTML5 中不再支持 <script> 元素的哪个属性（ ）

A. rel B. href C. type D. src

答案：C

解析：

HTML5 新增属性 async，之前被废弃的属性是 language，并且 type 是用来替代它的。在 HTML5 中，type 属性不再是必需的，默认值是“text/javascript”。

5.6 前端好考题（HTML5 布局）

第 294 道：实现布局，side 左边宽度固定，main 右边宽度自适应。

解析：

HTML 代码：

```

<div class="main">
    <div class="content"> 主栏 1 内容区内容区内容区内容区内容区内容区
    </div>
</div>
<div class="side"> 内容区内容区内容区内容区内容区内容区
</div>

```

CSS 代码：

```

.main {float: right; width:100%;margin-left:-220px;}
.content {margin-left:220px;}
.side{float:left;width:200px;}

```

第 295 道：实现布局，两列和左边宽度自适应，右边宽度固定为 200px。

解析：

```

<div style="width:90%; margin:0 auto;">
    <div style="width:200px; float:right;"> 这是右侧的内容 </div>
    <div style="margin-right:210px;"> 这是左侧的内容，宽度自适应 </div>
</div>

```

第 296 道：分别使用 2 个、3 个、5 个 div 画出一个大的红十字。

解析：

用 div 创建一个矩形，然后复制一个，以中心点旋转 90 度，两个合并就可以了。

用 div 创建一个矩形，然后复制一个，以中心点旋转 90 度，两个合并，然后在外面包一个空的 div 盒子。

红十字标志由五个正方形组成，代表五大洲，画一个正方形，然后复制成五个，再将五个正方形摆放好位置，组成红十字形状，圈选五个正方形之后，选择焊接选项，再填充红色（C:0，M:100，Y:100，K:0），去除边框颜色即可。

5.7 这些题，让你赢在起跑线上（关于 CSS3）

第 297 道：谈一下你对 CSS Reset 的了解情况。

解析：

CSS Reset，我们可以把它叫作 CSS 重设，也有人叫作 CSS 复位、默认 CSS、CSS 重置等。CSS 重设就是由于各种浏览器解释 CSS 样式的初始值有所不同，导致设计师在没有定义某个 CSS 属性时，不同的浏览器会按照自己的默认值来为没有定义的样式赋值，所以我们要先定义好一些 CSS 样式，来让所有浏览器都按照同样的规则解释 CSS，这样就能避免发生以下问题：

（1）标题部分采用的是 line-height 来实现标题中文字的间距，由于中英文有别，一般我们在实际项目中 h2 的大小设为 14px，高度设为 30px。而 bootstrap 或 normalize.css 面对的都是英文字体，所以它们默认设置的字体比中文的要大，且间距也比较大。

（2）根据我们常用的需求给 ul 添加两个 class 样式，一个为 has-style，顾名思义就是拥有列表样式，因为我们在一开始重置了没有样式，但是偶尔我们又确实需要前面的那个小圆点，所以就用这个 class 来还原；另一个为 inline-style，即 li 浮动，一般和 clearfix 结合使用以清除浮动。

（3）对于 normalize.css 不支持的 IE6，我们添加了 class 用以支持。因为 IE6 不支持属性选择器，所以 Form 表单元素的一些重置，我们在 normalize.css 的基础上添加了 class 用以支持。

（4）可以根据自己的需要，把用不到的 HTML5 或 CSS3 标签的那部分直接删掉，以精简。

5.8 领先别人一步（CSS3 选择器）

第 298 道：请解释浏览器是如何根据 CSS3 选择器选择对应元素的？

解析：

IE-CSS3.JS 下载页面的每一个样式文件并解析它的 CSS3 伪选择器。如果一个选择器被找到，它就会被替换为同名的 CSS class。比如：div:nth-child(2) 将会变成 div_IEcss-nth-child-2。接着，Robert Nyman 的 DOMAssistant 用于寻找匹配元素 CSS3 选择器的 DOM 节点，然后将相应的 CSS 类添加给它。最终，元素的样式表会被新的版本替代，实现用 CSS3 选择器对相应元素添加对应的样式。

5.9 从最陌生到最熟悉的题（切图）

第 299 道：你排 App 时是怎么切图的？

解析：

排版时切图可使用的软件很多，以下说一款 PS 切图。

打开 Photoshop，左边工具栏有个小刀形状的工具，叫“切片工具”，用小刀按照想要的样子在图片上切割，做好之后，点“文件→存储为 Web 所用格式”，选择路径后单击“确定”按钮，之后在所选的路径下有一个新的文件夹和一个 HTML 文件，文件夹里是切好的图片。

5.10 做好当下（定位相关）

第 300 道：关于元素的定位有哪些？其中对 z-index 样式的理解及 z-index 对不同浏览器默认的风格是什么？

解析：

position 属性规定元素的定位类型，position 属性的几个取值定义有：static、relative、absolute。

static：默认值。如果没有指定 position 属性，支持 position 属性的 HTML 对象都默认为 static，可以这么理解，把 HTML 页面看作一个文档流，源代码中各个标签的先后位置就是它们所对应的对象的呈现次序，所有取值为 static 的对象都按照所编写的 HTML 标签的顺序依次呈现。

relative：相对定位。这个属性值保持对象所在文档流中的位置，也就是说，它具有和 static 相同的呈现方式，它同样占有在文档流中的固定位置，后面的对象不会侵占或覆盖；与 static 属性值不同的是，它设置了 relative 的对象，可以通过 top、left、right、bottom 属性设定新的显示位置，这 4 个属性的取值是相对于文档流的前一个对象的，你可以自由设置这 4 个属性，偏移到的位置而不文档流中的其他对象产生任何影响，原来的页面呈现仍然会我行我素。

absolute: 绝对定位。和 **relative** 不同的是，这个属性值会将当前对象拖出文档流，后面的对象会占有原来的位置，也就是说，当前对象的呈现是独立显示的，但是它的位置在指定 **top**、**left**、**right**、**bottom** 任一属性之前仍是有继承性的，这时的 4 个属性的取值是相对于浏览器的，和文档流无关。属性值为 **absolute** 对象的 **z-index** 属性可以设置层叠显示的次序，它是直接有效的；而属性值为 **relative** 对象的 **z-index** 属性在设置时要小心，把当前对象的 **z-index** 设置为 -1 是不行的，在 Firefox 中它无法显示（注意，不是说浏览器有误，而是指如果父对象是根元素 **body**，那么 **z-index** 是无效的，任何 **z-index** 设置都不会显示在根元素之后，除了 IE 的解析 bug），必须设置为 0 以上，如果我们想让别的对象挡住它，只有将其他对象也设置 **position** 为 **relative**，并将 **z-index** 属性取一个比它大的值即可。

z-index 属性的默认值是 0。

元素可拥有负的 **z-index** 属性值，如 **z-index:-1**。

z-index 属性无继承性。

z-index 属性在 JavaScript 中使用语法：**object.style.zIndex="1"**。

几乎所有的主流浏览器都支持 **z-index** 属性。

z-index 在 IE 和 Firefox 下的默认值不同，在 IE 下 **z-index** 默认值为 0；在 Firefox 下其默认值为 **auto**。

第 6 章



不鸣则已，一鸣惊人 [HTML5+CSS3 中级面试题]

音频视频独揽群雄，盒布局助你收缩有度，代码纠错、优化让你游刃有余。
PHP 文件上传助你穿越时空，多列布局排列有序，稍有不慎，就会误入陷阱，切记！谨慎布局。

6.1 没有你们，我会不安（HTML5 音频与视频）

第 301 道：请用代码写出 <video> 标签的使用方法。

解析：

HTML5 的 video 标签是 HTML5 的一大特色，它有以下几种方法。

方法一：<video src="test.mp4"> 您的浏览器不支持 video 标签 </video>

方法二：<video><source src="test.3gp"> 您的浏览器不支持 video 标签 </video>

因为现阶段不同的浏览器支持的视频格式是不同的，当我们有多种格式的视频样式时，我们会用第二种写法来做兼容调试。

第 302 道：用于播放 HTML5 视频文件的正确 HTML5 元素是（ ）

A. <movie> B. <medio> C. <video>

答案：C

解析：

在 HTML5 中，“视频”标签 <video> 用于定义视频，比如电影片段或其他视频流。

第 303 道：请简单结合 HTML5，说说常用的流媒体视频格式有哪一种？

解析：

HTML5 常用的流媒体视频格式主要有 6 种：

(1) RealVideo 的 .rm 视频影像格式和 .ra 的音频格式，主要用来在低速率的网络上实时传输活动视频影像，可以根据网络数据传输速率的不同而采用不同的压缩比率，在数据传输过程中边下载边播放视频影像，从而实现影像数据的实时传送和播放。

(2) Microsoft Media Technology 的 .asf 格式，其播放器 Microsoft Media Player 已经与 Windows 捆绑在一起，不仅用于 Web 方式播放，还可以用于在浏

览器以外来播放影音文件。

(3) QuickTime 的 .qt、.mov 格式，用于保存音频和视频信息，具有先进的音频和视频功能，支持包括 Apple Mac OS、Microsoft Windows95/98/NT 在内的所有主流计算机操作系统。QuickTime 文件格式支持 25 位彩色，支持 RLC、JPEG 等领先的集成压缩技术，提供 150 多种视频效果。

(4) Flash 的 .swf 格式，具有体积小、功能强、交互能力好、支持多个层和时间线层等特点，故越来越多地应用到网络动画中。

(5) MetaStream 的 .mts 格式，它是一种新兴的网上 3D 开放文件标准（基于 Intel 构架），主要用于创建、发布及浏览可以防缩的 3D 图形和开发电脑游戏。

(6) Authorware 的 .aam 多媒体教学课件格式，这类课件利用 Shockwave 技术和 Web Package 软件，可以把 Authorware 生成的文件压缩为 .aam 和 .aas 流式文件格式播放；也可以用 Director 生成后，利用 Shockwave 技术改造为网上传输的流式多媒体课件。

6.2 终是拨开云雾见月明（弹性盒布局）

第 304 道：说说弹性盒布局。

解析：

引入弹性盒布局模型的目的是提供一种更加有效的方式来对一个容器中的条目进行排列、对齐和分配空白空间。即便容器中条目的尺寸未知或是动态变化的，弹性盒布局模型也能正常工作。在该布局模型中，容器会根据布局的需要，调整其中包含的条目的尺寸和顺序来最好地填充所有可用的空间。当容器的尺寸由于屏幕大小或窗口尺寸发生变化时，其中包含的条目也会被动态地调整。比如，当容器尺寸变大时，其中包含的条目会拉伸以占满多余的空白空间；当容器尺寸变小时，条目会缩小以防止超出容器的范围。弹性盒布局是与方向无关的。在传统的布局方式中，block 布局是把块在垂直方向从上到下依次排列的；而 inline 布局则是在水平方向来排列的。弹性盒布局并没有这样内在的方向限制，可以由开发人员自由操作。

第 305 道：了解弹性盒模型属性（Flexible Box）。

解析：

- box-orient：设置或检索弹性盒模型对象的子元素的排列方式。
- box-pack：设置或检索弹性盒模型对象的子元素的左右对齐方式。
- box-align：设置或检索弹性盒模型对象的子元素的上下对齐方式。
- box-flex：设置或检索弹性盒模型对象的子元素如何分配其剩余元素。
- box-flex-group：设置或检索弹性盒模型对象的子元素的所属组。

- `box-ordinal-group`: 设置或检索弹性盒模型对象的子元素的显示顺序。
- `box-direction`: 设置或检索弹性盒模型对象的子元素的排列顺序是否反转。
- `box-lines`: 设置或检索弹性盒模型对象的子元素是否可以显示换行。

6.3 心在天上，题在手上（HTML5 常见问题）

第 306 道：给如下 HTML5 代码片段

```
<input type="button" id="but1" value="but1" onclick=""?>
<input type="button" id="but2" value="but2"/>
```

中的 `but1` 添加适当的事件，使单击 `but1` 时，交换两个按钮。

解析：

```
<input type="button" id="but1" value="but1" onclick="this
.parentNode.insertBefore(document.getElementById('but2' ), this)"/>
<input type="button" id="but2" value="but2"/>
```

第 307 道：如何在 HTML5 页面中嵌入音频？

解析：

HTML5 包含嵌入音频文件的标准方式，支持的格式包括 `mp3`、`wav` 和 `ogg`。

```
<audio controls>
  <source src="1.mp3" type="audio/mpeg" />
  Your browser doesn't support audio embedding feature.
</audio>
```

第 308 道：HTML5 有哪些不同类型的存储？

解析：

HTML5 支持本地存储，速度快而且安全，在之前版本中是通过 `Cookie` 实现的。

有两种不同的对象可用来存储数据。

- 第一种：`localStorage`，适用于长期存储数据，浏览器关闭后数据不丢失；
- 第二种：`sessionStorage`，存储的数据在浏览器关闭之后自动删除。

6.4 多几分钟的准备，少几小时的麻烦（HTML 元素）

第 309 道：写出 Web 1.0 和 Web 2.0 常用的 HTML 标签。

解析：

Web 1.0 常用的 HTML 标签：`<u>`、``、`<i>`、`<table>`、`<p>`、`</br>`、`<nobr>`、`<hr>`。

Web 2.0 常用的 HTML 标签：`<article>`、`<header>`、`<nav>`、`<section>`、`<meter>`、

Web Storage 和 cookie 的区别：

Web Storage 的概念和 cookie 相似，区别是它是为了更大容量存储设计的。cookie 的大小是受限的，并且每次请求一个新的页面的时候 cookie 都会被发送过去，这样无形中浪费了带宽。另外 cookie 还需要指定作用域，不可以跨域调用。

除此之外，Web Storage 拥有 setItem、getItem、removeItem、clear 等方法，不像 cookie 需要前端开发者自己封装 setCookie、getCookie。但是 cookie 也是不可以或缺的：cookie 的作用是与服务器进行交互，作为 HTTP 规范的一部分而存在，而 Web Storage 仅仅是为了在本地“存储”数据而生的。

6.6 比上不足，比下有余（代码优化）

第 313 道：请简化这段 CSS 代码。

```
abc{
    padding-left:50px;
    padding-top:10px;
    padding-bottom:5px;
    padding-right:15px;
}
```

解析：

padding 属性的书写格式总共有以下几种。

- (1) padding:10px 的意思是上下左右值全是 10px。
- (2) padding:5px 10px 的意思是上下为 5px，左右为 10px。
- (3) padding:1px 2px 3px 4px 的意思是：上为 1px，右为 2px，下为 3px，左为 4px。
- (4) padding:5px 10px 7px 的意思是：上为 5px，左右为 10px，下为 7px。

padding 后面 4 个值定义的顺序分别为：上、右、下、左，而 padding-top 或者 padding-bottom 这种写法，只是单方面的定义了一个方向的值，这样写会增加 CSS 代码的长度，增加 CSS 样式的代码量，拖慢页面的加载速度。

第 314 道：请列出 CSS 代码的 7 个优化准则。

解析：

- (1) 使用简写。
- (2) 避免使用 hack。
- (3) 使用留白。
- (4) 移除多余的结构（frameworks）和重设（resets）。
- (5) 让 CSS 保证日后的维护。
- (6) 记录工作（标记向导和样式表向导）。

(7) 压缩使用。

6.7 非常可乐，非常选择（上传）

第 315 道：请写出可以实现上传且只能上传图片的标签。

解析：

HTML 自带一个上传文件控件：

```
<input type="file" name="uploadFile" />
```

再给 input 标签添加一个 accept="image/*" 的属性就只能上传图片了。

第 316 道：请写一段 JavaScript 代码，实现 input type="file" 文件上传实例的代码。

解析：

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
<style type="text/css">
._box
{
position: relative;
width: 119px;
height: 37px;
background-color: #53AD3F;
background-image: url(images/bg.png);
background-repeat: no-repeat;
background-position: 0 0;
background-attachment: scroll;
line-height: 37px;
text-align: center;
color: white;
cursor: pointer;
overflow: hidden;
z-index: 1;
}
._box input
{
position: absolute;
width: 119px;
height: 40px;
line-height: 40px;
```

```
font-size: 23px;
opacity: 0;
filter: "alpha(opacity=0)";
filter: alpha(opacity=0);
-moz-opacity: 0;
left: -5px;
top: -2px;
cursor: pointer;
z-index: 2;
}
</style>
<title>js 实现 input file 文件上传 /></title>
</head>
<body>
<form id="form1" runat="server" method="post"
enctype= "multipart/form-data">
<div>
<div class="_box">
<input type="file" name="_f" id="_f" />
选择图片
</div>
</div>
</form>
</body>
```

</html> 用一个不透明度为 0 的 <input type="file"/> 盖在要让用户可见的标签（或图片）上，让用户单击。

【总结】

用 width height line-height font-size 来控制 <input type="file"/> 右侧浏览器按钮的大小。用 left top(right、bottom) 来控制 <input type="file"/> 右侧浏览器按钮的位置，可以设置为负值。用 z-index 来设置它们的层覆盖关系。Form 必须有 enctype="multipart/form-data" 标记才能上传文件。

6.8 爱上面试的感觉（文本）

第 317 道：写出中英文强制换行的代码，以及文字超长显示的代码。

解析：

- (1) word-break:break-all; 只对英文起作用，以字母作为换行依据。
- (2) word-wrap:break-word; 只对英文起作用，以单词作为换行依据。
- (3) white-space:pre-wrap; 只对中文起作用，强制换行。

(4) white-space:nowrap; 强制不换行，都起作用。

(5) white-space:nowrap;overflow:hidden;text-overflow:ellipsis; 不换行，超出部分隐藏且以省略号形式出现。

第 318 道：在不指定特殊属性的情况下，哪几种 HTML 标签可以手动输入文本？（ ）（多选）

- A. `<textarea></textarea>` B. `<input type="text">`
C. `<input type="hidden">` D. `<div></div>`

答案：AB

解析：

`<textarea>` 可以输入复数行的文本输入框。写在 `<textarea> ~ </textarea>` 之间的文本会成为此文本输入框中内容的初始值。

`<input type="text">` 的默认值最基本的就是用 `value` 设置，例如 `<input type="text" value="默认值">`。

`<input type="hidden" name="参数名" value="这里填你的参数值">`。其实 `hidden` 只是将输入框隐藏了，里面可以默认赋值或通过 JavaScript 赋值，提交 Form 时跟其他的都是一样的。

`<div></div>` 标签只可以赋给定的值，不可以手动输入值。

第 319 道：如这段 HTML 代码：`<div>这是 div 中的内容</div>`，如何得到 `div` 元素中文本的内容？如何清空 `div` 元素中的文本？

解析：

`$("#div").text();` 获取中间的文本，不包括 HTML 标签。

`$("#div").html();` 获取中间的所有内容。

`$("#div").empty();` 清空 `div` 元素中的文本。

第 320 道：`font-size:62.5%`，解释一下如此设计字体大小的原因。

解析：

在网页设计中我们经常看见 `body{font-size: 62.5%;}` 这样的设置，这主要是为了方便 `em` 与 `px` 相互转换，`em` 的初始值为 `1em=16px`，显然这样的话，`1.2em=19.2px`，可是我们在设置时很少看见 `19.2px` 这样表示的大小，也就是在用 `px` 表示大小时数值是不带小数位的。当设置了 `body{font-size: 62.5%;}` 时，则 `1em=16px*62.5%=10px`，`1.2em=12px`，这样会使页面更精确。

第 321 道：`<div id="mydiv"><h2>通过 Ajax 改变文本</h2></div>`

`<button id="b01" type="button">改变文本</button>`

(1) 使用 `$(selector).load(url)` 把 HTML 文本内容改为 `XX.txt` 内容。

(2) 使用 `$.Ajax(option)` 把 HTML 文本内容改为 `XX.txt` 内容。

解析：

1. `$(document).ready(function(){`

```

    $("#b01").click(function(){
    $("#mydiv").load("/jQuery/xx.txt");
    });
2. $.Ajax({
    type: "POST",
    url: "xx.txt",
    success: function(msg){
        $('#id').txt(msg);
    }
});

```

6.9 你想摆谱，先干掉我（字体）

第 322 道：请指出以下结构中 A 标签内的字体颜色值。

```

<style>
  A{color:#ccc;}
  #contact a{color:#336699;}
  .safelink a{color:#eee;}
  H1 a{color:#eee;}
</style>

<div id="contact">
  <h1 class="safelink"><a
href="http://www.qdjh.com"> 前端江湖 </a></h1>
</div>

```

解析：

A 标签内的字体颜色是 #eee; 因为指定的优先级最高，越具体越强大。

第 323 道：请写出如下代码片段在不同浏览器 IE、Firefox、Chrome、Opera 中的字体颜色，并给出理由。

```

<style>
  #tip{
    color:blue;
    color:red\9;
    *color:black;
    _color:orange;
  }
</style>

```

解析：

```

#tip {
color: blue;/* 所有现代浏览器 */

```



```

color: red\9;/* 所有 IE 浏览器 */
*color: black;/*IE6、IE7 浏览器 */
_color: orange;/*IE6 浏览器 */
}

```

IE6: orange; IE6 以上: red\9; Firefox: blue; Chrome: blue; opera: blue;

第 324 道: <style type="text/css">

```

    #text{color:red;}
    P#text{color:brown;}
    P.text{color:green;}
    P{color:white;}
</style>
<p id="text" class="text"> 请说出我的颜色 </p>

```

请问 p 标签的文本颜色是什么?

解析:

p 标签的文本颜色是 brown，因为在选择器中，id 的优先级最高。

6.10 前端深处考题（边框背景）

第 325 道: CSS3 如何实现圆角、阴影效果?

解析:

(1) CSS3 实现圆角有两种方法:

第一种方法是添加背景图像，传统的 CSS 每个元素只能有一个背景图像，但是 CSS3 可以允许一个元素有多个背景图像。这样给一个元素添加 4 个 1/4 圆的背景图像，分别位于 4 个角上就可以实现圆角效果了。

```

.box {
    /* 首先，定义要使用的 4 幅图像为背景图 */
    background-image: url(/img/top-left.gif),
        url(/img/top-right.gif),
        url(/img/bottom-left.gif),
        url(/img/bottom-right.gif);
    /* 然后，定义不重复显示 */
    background-repeat: no-repeat,
        no-repeat,
        no-repeat,
        no-repeat;
    /* 最后，定义 4 幅图分别显示在 4 个角上 */
    background-position: top left,
        top right,
        bottom left,

```

```
        bottom right;
    }
}
```

第二种方法就简洁了，直接用 CSS 实现，不需要用图片。

```
.box {
    -moz-border-radius: 1em;
    -webkit-border-radius: 1em;
    border-radius: 1em;
}
```

(2) CSS3 的 box-shadow 属性可以直接实现阴影效果。

```
img {
    -webkit-box-shadow: 3px 3px 6px #666;
    -moz-box-shadow: 3px 3px 6px #666;
    box-shadow: 3px 3px 6px #666;
}
```

这个属性的 4 个参数是：垂直偏移、水平偏移、投影的宽度（模糊程度）、颜色。

第 326 道：box-sizing 属性有哪些，各代表什么含义？

解析：

box-sizing 属性允许以特定的方式定义匹配某个区域的特定元素。

content-box：padding 和 border 不被包含在定义的 width 和 height 之内。对象的实际宽度等于设置的 width、border、padding 以及 margin 之和，即 (Element width = width + border + padding + margin)，此属性表现为标准模式下的盒模型。

border-box：padding 和 border 被包含在定义的 width 和 height 之内。对象的实际宽度就等于设置的 width 值，即使定义有 border 和 padding 也不会改变对象的实际宽度，即 (Element width = width)，此属性表现为怪异模式下的盒模型。

6.11 有这些，更自信（多列布局）

第 327 道：宽度自适应三栏的布局方式有哪些？

解析：

宽度自适应三栏的布局方式，在这里我们介绍两种：

1. 绝对定位法

左右两栏采用绝对定位，分别固定于页面的左右两侧，中间的主体栏用左右 margin 值撑开距离，于是实现了三栏自适应布局。

2. 自身浮动法

此方法代码最简单，应用了标签浮动跟随的特性。左栏左浮动，右栏右浮

动，主体直接放后面，就实现了自适应。

第 328 道：说说 CSS3 的多列布局。

解析：

CSS3 中新出现的多列布局（multi-column）是传统 HTML 网页中块状布局模式的有力扩充。这种新语法能够让 Web 开发人员轻松地让文本呈现多列显示。我们知道，当一行文字太长时，读者读起来比较费劲，有可能读错行或读串行。人们的视点从文本的一端移到另一端，然后换到下一行的行首，如果眼球移动浮动过大，注意力就会减退，容易读不下去。所以，为了最大效率地使用大屏幕显示器，页面设计中需要限制文本的宽度，让文本按多列呈现，就像报纸上的新闻排版一样。

6.12 总有些考题念念不忘（多列显示样式）

第 329 道：写出三列布局的 CSS，其中 HTML 为：`<div id="A"></div><div id="B"></div><div id="C"></div>`，要求不改变 HTML 结构，用 CSS 分别实现，按照 ABC 排列、按照 BAC 排列、按照 CBA 排列。其中 AB 为定宽，尽量让 B 为自适应宽度，不能使用 CSS hack。

解析：

CSS 三列布局走出 HTML 布局阴影，两端固定宽度，中间自适应结构，以下代码仅供参考。

HTML 代码：

```
<div class="wrap">
  <div class="main">
    <div id="A"></div>
  </div>
  <div id="B"></div>
  <div id="C"></div>
</div>
```

CSS 代码：

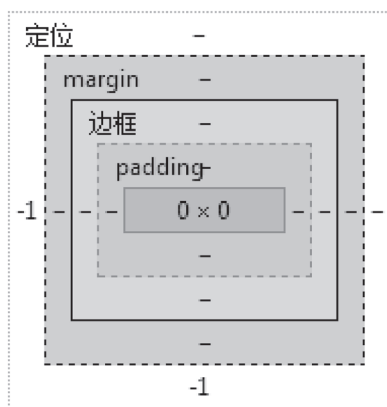
```
.wrap{width: 500px;border: 1px solid;overflow: hidden;}
.main{float: left;width: 100%;}
#A{height: 50px;margin: 0 150px;background-color: #f60;}
#B{float:left;width:150px;height:50px;margin-left:-100%;background-color: #6f0;}
#C{float:left;width:150px;height:50px;margin-left:-150px;background-color: #06f;}
```

6.13 深入每道题的世界（盒布局）

第 330 道：画出 CSS 的盒模型。

解析：

盒模型是 CSS 中的重要概念之一，它是所有布局控制的基础，在 CSS 标准中，一个盒模型包括 4 个区，分别是内框、内边距、边框和外边距。在指定一个元素的大小时，就是根据盒模型中各个部分的大小来决定的。CSS 的盒模型如下图所示。



CSS 的盒模型

第 7 章



运筹帷幄，决胜千里 [HTML5+CSS3 高级面试题]

永不过期的模式相关，闯全关的媒体查询，适应于任何恶劣的战斗环境，让你更强大的媒体调用标签，Canvas 统筹布局，变化多端，有七十八般武艺让你眼花缭乱。

7.1 就这些，永不过期（模式）

第 331 道：什么是 DOCTYPE？如何触发严格模式与混杂模式，这两种模式，它们有何区别？

解析：

DOCTYPE（是 DOCument TYPE 的简写，即文档类型）是一组机器可读的规则，它们指示 (X)HTML 文档中允许有什么，不允许有什么。DOCTYPE 正是用来告诉浏览器使用哪种 DTD，一般放在 (X)HTML 文档开头表示声明，用以告诉其他人这个文档的类型风格。

触发：根据不同的 DTD 触发，如果没有声明，那么默认为混杂模式。

区别：严格模式是浏览器根据 Web 标准去解析页面，是一种要求严格的 DTD，不允许使用任何表现层的语法，如 `
`，混杂模式则是一种向后兼容的解析方法。

第 332 道：Strict 与 Transitional 这两种模式有何意义？如何声明 HTML5？

解析：

Strict 是需求最苛刻的 XHTML 规范，但是它提供了最干净的结构化标记。Strict 编码独立于任何定义外观的标记语言。它使用层叠样式表（CSS）来控制表示外观。这种与表示相独立的结构模式，使得 XHTML Strict 能够相当灵活地在不同的设备上显示。而其控制和表示对 CSS 的依赖，对于程序员来说又比较麻烦，因为如果想要在那些不能识别样式表的设备或浏览器中显示 Web 内容，它并不是一个好的选择。

Transitional 是更加宽容的规范。Strict 完全将结构与表示分离，而 Transitional 允许使用标签来控制外观。它的目的是要在允许用标记来控制表示的 HTML 页面和二者完全分离的 XHTML Strict 之间架起桥梁。它最大的好处是

克服了 Strict 对 CSS 的依赖。Transitional 页面对于使用旧式浏览器或不能识别样式表的用户来说也是可以访问的。

HTML5 的声明: <!DOCTYPE html>

7.2 这些题，让你前端技艺更高一筹（HTML5 页面）

第 333 道：如何触发页面 reflow, repaint ?

解析：

除了页面在首次加载时必然要经历该过程之外，还有以下行为会触发这个行为：

- (1) DOM 元素的添加、修改（内容）、删除（reflow+ repaint）;
- (2) 仅修改 DOM 元素的字体颜色（只有 repaint，因为不需要调整布局）;
- (3) 应用新的样式或者修改任何影响元素外观的属性;
- (4) resize 浏览器窗口、滚动页面;
- (5) 读取元素的某些属性（offsetLeft、offsetTop、offsetHeight、offsetWidth、scrollTop/Left/Width/Height、clientLeft/Left/Width/Height、getComputedStyle()、currentStyle(in IE)）。

第 334 道：如何将一个表单中的值同时提交到两个页面？写出代码。

解析：

```
<script>
    function send_2page_f(formname,page1,target1,page2,target2)
    {
        with(eval("document."+formname))
        {
            action=page1
            target=target1
            submit()
            action=page2
            target=target2
            submit()
        }
    }
</script>
<form name="form">
    <input type="button"
    onclick='send_2page_f("form","01.
    asp","01","02.asp","02")' value="提交">
</form>
```

7.3 考题中的钉子户（Canvas 的使用）

第 335 道：哪个 HTML5 内建对象用于在画布上绘制？（ ）

A. getContent B. getContext C. getGraphics D. getCanvas

答案：B

解析：

getContext() 方法返回一个用于在画布上绘图的环境。

Graphics 是 Java 绘图的核心类，它可以支持两种绘图方式：一种是基本的绘图，如画线、矩形、圆等；另一种是画图像，主要用于动画制作。

第 336 道：HTML5 中的 <canvas> 元素用于（ ）

A. 显示数据库记录 B. 操作 MySQL 中的数据
C. 绘制图形 D. 创建可拖动的元素

答案：C

解析：

Canvas API（画布）用于在网页实时生成图像，并且可以操作图像内容，基本上它是一个可以用 JavaScript 操作的位图（bitmap）。

Canvas 对象表示一个 HTML 画布元素 <canvas>。它没有自己的行为，但是定义了一个 API 支持脚本化客户端绘图操作。

你可以直接在该对象上指定宽度和高度，但是，其大多数功能都可以通过 CanvasRenderingContext2D 对象获得。这是通过 Canvas 对象的 getContext() 方法并且把直接量字符串“2d”作为唯一的参数传递给它而获得的。

<canvas> 标记在 Safari 1.3 中引入，在制作此参考页时，它在 Firefox 1.5 和 Opera 9 中也得到了支持。在 IE 中，<canvas> 标记及其 API 可以使用位于 excanvas.sourceforge.net 的 ExplorerCanvas 开源项目来模拟。

第 337 道：写出在 Canvas 中画圆的代码，直径为 150px，边框宽度为 5px。

解析：

```
<!DOCTYPE html>
<html>
<body>
    <canvas id="myCanvas" width="200" height="200"
    style="border:2px solid blue;">
        您的浏览器不支持。
    </canvas>
    <script type="text/javascript">
        var c=document.getElementById("myCanvas");
        var cxt=c.getContext("2d");
        cxt.fillStyle="#FF0000";
        cxt.beginPath();
        cxt.arc(100,100,75,0,Math.PI*2,true);
```

```

        cxt.closePath();
        cxt.fill();
    </script>
</body>
</html>

```

7.4 点面试，闯全关（媒体查询）

第 338 道：<meta name="viewport" content="width=device-width, minimum-scale=1.0, maximum-scale=1.0"/>

解释这个标签行的作用。

解析：

网页手机 Wap2.0 网页的 head 里加入这条元标签，在 iPhone 的浏览器中，页面将以原始大小显示，并不允许缩放。

width —— viewport 的宽度；

height —— viewport 的高度；

initial-scale —— 初始的缩放比例；

minimum-scale —— 允许用户缩放到的最小比例；

maximum-scale —— 允许用户缩放到的最大比例；

user-scalable —— 用户是否可以手动缩放。

第 339 道：写出 media type 的几种使用方法。

解析：

方法一：<link href="style.css" media="screen print"...

方法二：<?Xml-stylesheet media="screen " href="style.css"...

方法三：@import url("style.css") screen;

方法四：<style media="screen" >

@import url("style.css")

</style>

方法五：media screen{

Selector {rules}

}

7.5 面试一大坎儿（浏览器缓存与本地储存）

第 340 道：HTML5 的存储类型有哪些？

解析：

HTML5 支持本地存储，在之前版本中是通过 Cookie 实现的。HTML5 本地存储速度快而且安全。

HTML5 有两种不同的对象可用来存储数据。

localStorage：适用于长期存储数据，浏览器关闭后数据不丢失；

sessionStorage：存储的数据在浏览器关闭后自动删除。

第 341 道：HTML5 应用程序缓存为应用带来什么优势？

解析：

应用程序缓存为应用带来三个优势。

- 离线浏览——用户可在应用离线时使用它们；
- 速度——已缓存资源加载得更快；
- 减少服务器负载——浏览器将只从服务器下载更新过或更改过的资源

7.6 这些让你更强大（媒体调用标签）

第 342 道：请写出可以调用手机照相机的标签。

解析：

```
<input type="file" accept="audio/*;capture=microphone"/>
```

第 343 道：标签代码调用大全。

解析：

关键描述调用标签：`<meta name="keywords" content="{dede:field name='keywords'}/>`

`<meta name="description" content="{dede:field name='description' function='html2text(@me)'/}>`

模板路径调用标签：`{dede:field name='templeturl'}`

网站标题调用标签：`{dede:global name='cfg_webname'}`

栏目导航调用标签：` 首页 `

`{dede:channel type='top' row='8' currentstyle="<li class='thisclass'>~typename~ "}`

`[field:typename/] `

`{/dede:channel}`

指定栏目调用标签：`{dede:onetype typeid='ID'}[field:typename /]{/dede:onetype}`

频道栏目调用标签：`{dede:channel type='self'}[field:typename/]{/dede:channel}`

友情链接调用标签：`{dede:flink row='24' linktype=2/}`

网站版权调用标签: {dede:global name='cfg_powerby'/}
 网站备案调用标签: {dede:global name='cfg_beian'/}
 当前栏目名称调用标签: {dede:field name='typename'/}
 当前位置调用标签: {dede:field name='position'/}
 列表文章调用标签: {dede:list pagesize='8'}{/dede:list}
 栏目链接调用标签: [field:typelink function='str_replace("a ","a class=ulink ",@me)'/]
 作者链接调用标签: [field:writer /]
 列表单击调用标签: [field:click/]
 列表评论调用标签: [field:postnum/]
 查阅全文调用标签: 查阅全文 ...

7.7 脑若一动，题就千行（HTML、CSS 综合）

第 344 道：简述一下什么是内容与表现分离？CSS 都有哪些调用方式？

解析：

页面显示的是内容和相应的表单，而 CSS 可以很好地将这些内容进行分块组装后，再按照整体的布局进行调整，可以在后期的改版等方面能够不动前面的内容，只是更改布局及背景、图片、表现形式等，这样更便于优化，所以才进行了分离。其实不分离的话，就是说每个页面你都需要在里面嵌入 CSS 内部样式，如果你想每个页面的样式都一样，那么每个页面的样式都要重新 COPY 一遍，对于后面的修改来说更改量太大了，而用 CSS 和页面分离的方式可以只修改一个 CSS 文件来使整个网站的样式进行优化，便于修改，因为 HTML 代码当中已经不包含样式代码了，样式文件被单独放到了一块，可以单独调用，也可以包含在 <style type="text/css"></style> 标记里。总之，和 HTML 代码不在交叉了。

CSS 有四种调用方式：内行样式、内嵌样式、链接样式、导入样式。

第 345 道：HTML 和 CSS 的关系。

解析：

学习 Web 前端开发基础技术需要掌握：HTML、CSS、JavaScript 语言。下面我们就来了解下这三门技术都是用来实现什么的。

(1) HTML 是网页内容的载体。内容就是网页制作者放在页面上想要用户浏览的信息，可以包含文字、图片、视频等。

(2) CSS 样式是表现，就像网页的外衣。比如，标题字体、颜色变化，或为标题加入背景图片、边框等，所有这些用来改变内容外观的东西称之为表现。

(3) JavaScript 是用来实现网页上的特效效果。如：鼠标滑过表格时背景颜色改变，还有焦点新闻（新闻图片）的轮换等。可以理解，有动画的、有交互的一般都是用 JavaScript 来实现的。

7.8 有了我就知足吧（兼容问题）

第 346 道：Box{float: left;margin: 10px;} 前面的样式有何兼容性问题？
如何解决？

解析：

在 IE6 下会出现双倍间距问题，这是一个 IE6 都存在的 Bug。

解决办法是在 Box 中加 display: inline。

第 347 道：如何解决 IE6 下 div 错位的问题，以及 IE7、IE8 样式不兼容问题？

解析：

1. IE6 下 div 错位的问题

在 IE7、IE8 下采用“FLOAT：LEFT”的 div 都没问题，但是在 IE6 下却会向下移动，出现空白，这是因为 IE6 采用的内核默认把 div 之间的距离增加了 3~5 个 px，所以，可在下移的 div 的 style 里增加 "margin-left:-5px;" 或者更小。

2. 解决 IE7、IE8 样式不兼容问题

在页面中加入 HTTP meta-tag:

```
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7" />
```

只要 IE8 读到这个标签，它就会自动启动 IE7 兼容模式，保证页面完整展示。

7.9 一直在寻找，直到遇见你（响应式布局）

第 348 道：谈谈你对响应式布局的看法？

解析：

响应式布局有缺点，也有优点。

优点：面对不同分辨率设备，灵活性强，能够快捷地解决多设备显示适应问题。

缺点：兼容各种设备时所需工作量大、效率低下、代码累赘，会隐藏无用的元素，加载时间延长，其实这是一种折衷性质的设计解决方案，由于多方面因素影响而达不到最佳效果，在一定程度上改变了网站原有的布局结构，会出现用户混淆的情况。

第 349 道：你知道哪几个响应式框架？

解析：

响应式 Web 设计的运用越来越广泛。因此，作为一名 Web 开发人员和前端设计师，我们有必要了解更多的有关响应式设计的工具和资源，使其更容易为我们创建响应式的 Web 站点。希望以下这些工具能帮你更快速地创建一个响应式 Web 站点。

- (1) Titan Framework。
- (2) Responsive Grid System。
- (3) Ingrid。
- (4) Gumby。
- (5) Susy: Responsive grids for Compass。
- (6) Responsive Grid System。
- (7) Foundation 3。
- (8) Less + Framework。

7.10 一直在寻觅的考题（关于浏览器）

第 350 道：如何触发浏览器的 Quirks Mode？在该模式下的盒子模型会有什么表现？

解析：

(1) 没写 DOCTYPE，触发浏览器的 Quirks Mode。

(2) 加 XML 声明，在 IE6 下触发：

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<!DOCTYPE ...>
```

(3) 在 XML 声明和 XHTML 的 DOCTYPE 之间加入 HTML 注释，可在 IE7 下触发 `<?xml version="1.0" encoding="utf-8"?>`

```
<!-- keep IE7 in Quirks Mode -->
```

```
<!DOCTYPE ...>
```

(4) `<!-->` 放在 `<!DOCTYPE` 的前面。

(5) 写了 DOCTYPE，但不在文档的第一行。实验证明，在 DOCTYPE 之前有任何非空字符都会触发 IE6 的怪异模式，在 IE7 下，DOCTYPE 之前有 XML 的文档声明并不触发，但是在 DOCTYPE 和 XML 文档声明之间有任何非空字符仍然会触发。

Quirks Mode 就是浏览器为了兼容很早之前针对旧版本浏览器而设计的，并未严格遵循 W3C 标准的网页而产生的一种页面渲染模式。

在 Quirks Mode 中：`width` 则是元素的实际宽度，`内容宽度 = width - (margin-`

left+margin-right+padding-left+padding-right+border-left-width+border-right-width)。

table 的 CSS 属性 font-size 是不会继承父级元素的，需要专门设置一下。也就是说，table 中的字体大小不会继承父级元素字体的大小。

7.11 无法轻描淡写的考题（PC、移动）

第 351 道：PC 端和移动端哪个做得多，PC 端和移动端有什么区别？你觉得 PC 端和移动端哪个比较好一些？

解析：

本题从以下 4 个方面来分析。

(1) 操作尺度。PC 上鼠标的操作尺度比较小，单击是一件准确的事情，而移动端触屏的操作尺度就比较大，单击误差大，所以元素必须做大一些，也不支持 hover 事件。这一点，淘宝网页的 PC 版和手机版是个非常好的例子。在 PC 版的淘宝页面中，有些小按钮能放下的功能，移动版就必须另弹界面让用户详细输入。

(2) 界面布局。移动端屏幕相对窄小，一般是单列，最多也只能是双列 + 响应式。PC 端屏幕宽大，布局可以灵活一些。

(3) 与 Native App 互动。Web App 与移动 App 间的互动，大多数 PC 网站没有，因为大多数 PC 网站没有对应的专门程序（不过也有淘宝或 QQ 等例外）。例如，淘宝手机版调用淘宝手机客户端，PC 版淘宝调用阿里旺旺聊天工具。

(4) 开发工具。移动端开发有一些 jQuery for mobile 一类的库是专用的，不用于 PC 端。反过来也一样，有些 PC 端的工具在移动端不好用。PC 应用开发更关注的是后台、大数据、算法类的，移动开发更关注的是如何更好地交互、体验。

泛泛而论，感觉还是移动开发更好，因为和用户有更直接的接触，而且以后应用会更广泛，移动设备还会延展人的感觉、触觉等，如网上买衣服，可以通过移动终端感受布料。移动毕竟是大势所趋，以后会越来越多。

第 352 道：你排 PC 端页面的时候兼容哪些浏览器？

解析：

浏览器不断更新，人们所熟知、使用的浏览器早已不只有电脑自带的浏览器了，所以我们排 PC 端的页面，要尽可能多的兼容更多浏览器。浏览器大体分为两种，IE 内核浏览器和非 IE 内核浏览器。

IE 内核浏览器有：360、傲游、搜狗、世界之窗、腾讯 TT；非 IE 内核浏览器有：Firefox、Opera、Safari、Chrome。一般排 PC 主要兼容 IE6、IE7、IE8，Firefox5+，Chrome，Safari 等浏览器。

7.12 最难懂的题给真心的你（HTML5 效果）

第 353 道：你能用 HTML5 做一些酷炫的效果吗？

解析：

本题旨在考察面试者对 HTML5 的了解和使用，只要如实地说出自己做过的一个效果就好。但要注意，介绍时要描述清楚效果，以及实现该效果所运用到的属性方法等。

下面是一个简单的例子。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>一款带有发光动画的 HTML5 表单</title>

<script>
  $(function(){
    var $form_inputs = $('form input');
    var $rainbow_and_border = $('.rain, .border');
    /* Used to provide loping animations in fallback mode */
    $form_inputs.bind('focus', function(){
      $rainbow_and_border.addClass('end').removeClass
('unfocus start');
    });
    $form_inputs.bind('blur', function(){
      $rainbow_and_border.addClass('unfocus start').
removeClass('end');
    });
    $form_inputs.first().delay(800).queue(function() {
      $(this).focus();
    });
  });
</script>
</head>
<body id="home">
  <div class="rain">
    <div class="border start">
      <form>
        <label for="email">Email</label>
        <input name="email" type="text"
placeholder="Email"/>
        <label for="pass">Password</label>
```

```

        <input name="pass" type="password"
placeholder="Password"/>
        <input type="submit" value="LOG IN"/>
    </form>
</div>
</div>
<div style="width:640px;margin:10px auto 20px auto;
padding:0 0 0 380px;overflow:hidden">
<!-- JiaThis Button BEGIN -->
<div id="jiathis_style_32x32">
    <a class="jiathis_button_qzone"></a>
    <a class="jiathis_button_tsina"></a>
    <a class="jiathis_button_tqq"></a>
    <a class="jiathis_button_renren"></a>
    <a class="jiathis_button_kaixun001"></a>

    <a class="jiathis_counter_style"></a>
</div>

</div>
<div style="font-size:18px;text-align:center;padding:0 0
20px 0">

</div>
<!-- JiaThis Button END -->
    </body>
</html>

```

7.13 从此，面试不重来（控件相关）

第 354 道：哪种输入类型定义滑块控件？（ ）

A. Seacher B. Controls C. Slider D. Range

答案：D

解析：

Range 定义带有 Slider 控件的数字字段。

Controls 属性规定浏览器应该为视频提供播放控件。

Slider 控件是由滑块与滑动条组成的。使用 Slider 控件，可以计算出滑块在滑动工程中占整个滑动条的比例。如果滑动条的整体长度为 100，则滑动的范围

就是 0 ~ 100。

第 355 道：哪种输入类型用于定义周和年控件（无时区）？（ ）

A. Date B. Week C. Year

答案：B

解析：

Date 用于定义时间。

Week 类型允许你选择周和年 Select a week: >。

Year 用于定义年。

第 8 章



见多识广，独霸一方 [HTML5+CSS3 终极面试题]

包罗万象的 HTML5 的 API 让你乘风破浪，CSS 动画让你随时变换人物特征，Geolocation 地理让你随时获取敌人的具体方位，将之一网打尽；workers 多线程处理默默无闻，无私奉献。

8.1 这些题必须认真对待（HTML5 应用程序缓存）

第 356 道：请写出 localStorage 对象的常用方法。

解析：

1. 存储：localStorage.setItem(key, value)

如果 key 存在，那么更新 value。

2. 获取：localStorage.getItem(key)

如果 key 不存在，那么返回 null。

3. 删除：localStorage.removeItem(key)

一旦删除，key 对应的数据将会全部删除。

4. 全部清除：localStorage.clear()

某些时候使用 removeItem 逐个删除太麻烦，可以使用 clear，执行的后果是会清除所有 localStorage 对象保存的数据。

5. 遍历 localStorage 存储的 key

.length 数据总量，例如：localStorage.length;

.key(index) 获取 key，例如：var key=localStorage.key(index);

6. 存储 JSON 格式数据，json.stringify(data)

将一个对象转换成 json 格式的数据串，返回转换后的串，Json.parse(data) 将数据解析成对象，返回解析后的对象。

第 357 道：HTML5 引入了应用程序缓存，如何启用应用程序缓存？

解析：

如需启用应用程序缓存，请在文档的 <html> 标签中包含 manifest 属性：每个指定了 manifest 的页面在用户对其访问时都会被缓存。如果未指定 manifest 属性，则页面不会被缓存（除非在 manifest 文件中直接指定了该页面）。

8.2 这些题“包罗万象”（HTML5 常见 API）

第 358 道：你知道的网页制作会用到的图片格式有哪些？

解析：

png-8、png-2、jpeg、gif、svg。

但是上面的那些都不是面试官想要的满意答案。面试官希望听到是 Webp、Apng（考察面试人员是否有关关注新技术、新鲜事物）。

WebP 格式是谷歌（Google）开发的一种旨在加快图片加载速度的图片格式。图片压缩体积大约只有 jpeg 的 2/3，并能节省大量的服务器带宽资源和数据空间。Facebook、ebay 等知名网站已经开始测试并使用 WebP 格式。

在图片质量相同的情况下，WebP 格式图像的体积要比 jpeg 格式的图像小 40%。

8.3 再深的题海，也能乘风破浪（HTML5 数据存储）

第 359 道：请描述一下 cookie、sessionStorage 和 localStorage 的区别。

解析：

cookie 数据始终在同源的 HTTP 请求中携带（即使不需要），即 cookie 在浏览器和服务端间来回传递。而 sessionStorage 和 localStorage 不会自动把数据发给服务器，仅在本地保存。cookie 数据还有路径（path）的概念，可以限制 cookie 只属于某个路径下。

存储大小限制也不同，cookie 数据不能超过 4KB，同时因为每次 HTTP 请求都会携带 cookie，所以 cookie 只适合保存很小的数据，如会话标识。sessionStorage 和 localStorage 虽然也有存储大小的限制，但比 cookie 大得多，可以达到 5MB 或更大。

数据有效期不同，sessionStorage：仅在当前浏览器窗口关闭前有效，自然也就不可能持久保持；localStorage：始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据；cookie 只在设置的 cookie 过期时间之前一直有效，即使窗口或浏览器关闭。

作用域不同，sessionStorage 不在不同的浏览器窗口中共享，即使是同一个页面；localStorage 在所有同源窗口中都是共享的；cookie 也是在所有同源窗口中都是共享的。Web Storage 支持事件通知机制，可以将数据更新的通知发送给监听者。Web Storage 的 API 接口使用更方便。

第 360 道：HTML5 提供了哪两种在客户端存储数据的新办法？有何区别？

解析：

一种是 localStorage，另一种是 sessionStorage，它们的区别如下。

- localStorage: 没有时间限制的数据存储。
- sessionStorage: 针对一个 session 的数据存储。

8.4 这些题让你相信能，就能！（HTML5 编辑 API）

第 361 道：文档类型的作用是什么？你知道多少种文档类型？

解析：

Web 世界中存在许多不同的文档。只有了解文档的类型，浏览器才能正确地显示文档。

HTML 也有多个不同的版本，只有完全明白页面中使用的确切 HTML 版本，浏览器才能完全正确地显示出 HTML 页面。这就是 <!DOCTYPE> 的用处。从 Web 诞生早期至今，已经发展出多个 HTML 版本，我们常用的声明有：

HTML5: <!DOCTYPE HTML>

HTML 4.01 : <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/HTML4/loose.dtd">

XHTML 1.0 : <!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/XHTML1/DTD/XHTML1-Transitional.dtd">

8.5 搞清楚这些让你“屌炸天”（CSS 动画）

第 362 道：请使用 CSS3 的 animation 创建一段动画，该动画的过程为：100px×100px 的图片以中心为圆点，顺时针原地旋转 3 秒，然后开始水平滚动 7 秒，所有动画均为匀速，要求适配浏览器为 Chrome 和 Safari。

解析：

```
@-webkit-keyframes ha{
    0% {
        -webkit-transform: rotateZ(0deg);
    }
    30% {
        -webkit-transform: rotateZ(200deg);
    }
    100%{
        -webkit-transform: translate(200px,
        0px);rotateZ(160deg)
    }
}
```

```

    }
    .box:hover{
        -webkit-animation: ha 10s linear infinite;
    }

```

第 363 道：CSS3 中分别有哪几种动画模式，分别有什么特点？

解析：

CSS3 中的动画有 Transitions、Transforms 和 Animations 三种模式。

Transitions 特点：平滑地改变 CSS 的值。

Transforms 特点：基本的变换，主要实现拉伸、压缩、旋转、偏移等。

Animations 特点：适用于 CSS2、CSS3。

第 364 道：你以前做过动画吗？简单描述一下。

解析：

动画主要分为两种，一种是动画（animation），另一种是过渡动画（transition）。

1. animation

- animation 与 transition 有几个相同的属性，duration、timing-function、delay；
- animation-name：animation 需要指定动画的名字，这个是用来在 @keyframes 中指定执行动画的样式；
- animation-duration：动画的执行时间，单位为 s（秒）；
- animation-timing-function：动画类型，取值与 transition 相同；
- animation-delay：动画延迟时间，单位为 s（秒）；
- animation-iteration-count：动画执行的次数，这个属性就克服了 transition 局限中的第二条，取值可以是数字，指定具体循环几次，也可以设置关键字“infinite”无限循环；
- animation-direction：动画在循环执行时是否反方向，取值“normal”正常方向、“alternate”正常与反向交替；
- animation-play-state：设置执行中动画的状态，取值“running”、“paused”；
- animation-fill-mode：设置动画执行时间之外的状态，取值“none”（默认，不设置）、“forwards”（动画结束时的状态）、“backwards”（动画开始时的状态）、“both”（开始或结束时的状态）。

2. transition

- transition-property：参与过渡的属性，比如 height、width 等 CSS 样式，所有样式用关键字“all”，没有样式需要过渡使用“none”；
- transition-duration：过渡持续的时间，单位是 s（秒），比如 width 从 100px 过渡到 200px 排序需要的时间；
- transition-timing-function：过渡的动画类型，有以下几种取值。
- linear：线性过渡，等同于贝塞尔曲线（0.0, 0.0, 1.0, 1.0）；

- ease: 平滑过渡，等同于贝塞尔曲线 (0.25, 0.1, 0.25, 1.0);
- ease-in: 由慢到快，等同于贝塞尔曲线 (0.42, 0, 1.0, 1.0);
- ease-out: 由快到慢，等同于贝塞尔曲线 (0, 0, 0.58, 1.0);
- ease-in-out: 由慢到快再到慢，等同于贝塞尔曲线 (0.42, 0, 0.58, 1.0);
- cubic-bezier(<number>, <number>, <number>, <number>): 特定的贝塞尔曲线类型，4 个数值需在 [0, 1] 区间内;
- transition-delay: 动画延迟执行的时间;
- transition 同 background 属性一样，有缩写和拆分两种方式，就是说，你可以把每个属性都加到 transition 后。

缩写如: transition: border-color .5s ease-in .1s;

也可以把这些属性拆分，分别设置在个属性之后，上边的缩写拆分后如下:

- transition-property: border-color;
- transition-duration: .5s;
- transition-timing-function: ease-in;
- transition-delay: .1s。

如果需要对个样式设置过渡动画，可以用逗号分开。

缩写如:

- transition: border-color .5s ease-in .1s, color .3s ease .2s;

拆分写法如:

- transition-property: border-color, color;
- transition-duration: .5s, .3s;
- transition-timing-function: ease-in, ease;
- transition-delay: .1s, .2s。

在拆分写法中，如果某个属性是多组共用的，则不需要分别设置，比如：
transition-duration: .5s, .5s; ---> transition-duration: .5s;

但是 transition 有几个很大的局限:

- (1) transition 需要事件触发，所以没法在网页加载时自动发生。
- (2) transition 是一次性的，不能重复发生，除非一再触发。
- (3) transition 只能定义开始状态和结束状态，不能定义中间状态，也就是说，只有两个状态。
- (4) 一条 transition 规则，只能定义一个属性的变化，不能涉及多个属性。所以就诞生了 animation 属性，相对 transition 略显复杂，但是功能更强大。

8.6 前端面试独家宝贝 (cache 机制)

第 365 道：浏览器的 cache 机制是怎样的？如果想让某个文件不 cache，如

何处理？

解析：

cache 机制，即浏览器缓存机制（cache-control），指明当前资源的有效期，控制浏览器是直接从浏览器缓存取数据，还是重新发请求到服务器取数据。cache-control 是关于浏览器缓存的最重要的设置，因为它覆盖其他设置，比如 Expires 和 Last-Modified。另外，由于浏览器的行为基本相同，这个属性是处理跨浏览器缓存问题的最有效的方法。

Expires 头部字段提供一个日期和时间，响应在该日期和时间后被认为失效。失效的缓存条目通常不会被缓存（无论是代理缓存，还是用户代理缓存）返回，除非首先通过原始服务器（或者拥有该实体的最新副本的中介缓存）验证。（注意：cache-control max-age 和 s-maxage 将覆盖 Expires 头部。）Expires 字段接收以下格式的值：“Expires: Sun, 08 Nov 2009 03:37:26 GMT”。如果查看内容时的日期在给定的日期之前，则认为该内容没有失效，并从缓存中提取出来；反之，则认为该内容失效，缓存将采取一些措施。

8.7 前端大牛的看家本事（workers 多线程处理）

第 366 道：什么是 web.worker？如何创建一个简单的 web.worker 实例？

解析：

web.worker 是在后台运行的 JavaScript，独立于其他脚本，不会影响页面的性能。你可以继续执行你想做的任何事情：单击、选取等，同时 web.worker 就在后台运行。

以下是一个简单的 web.worker 实例：

```
<!DOCTYPE html>
<html>
<head>
<title>Big for loop</title>
<script>
    function bigLoop(){
        for (var i = 0; i <= 10000000000; i += 1){
            var j = i;
        }
        alert("Completed " + j + "iterations" );
    }
    function sayHello(){
        alert("Hello sir..." );
    }
}
```

```

    </script>
</head>
<body>
    <input type="button" onclick="bigLoop();" value="Big Loop" />
    <input type="button" onclick="sayHello();" value="Say Hello" />
</body>
</html>

```

8.8 一入考题深似海，从此面试是浮云（Geolocation 地理位置）

第 367 道：在 HTML5 中，哪个方法用于获得用户的当前位置？（ ）

A. getPosition() B. getCurrentPosition() C. getUserPosition()

答案：B

解析：

getCurrentPosition(successCallback, errorCallback, positionOptions)

successCallback：表示调用 getCurrentPosition 函数成功以后的回调函数，该函数带有一个参数，对象字面量格式，表示获取到的用户位置数据。该对象包含两个属性：coords 和 timestamp。其中，coords 属性包含以下 7 个值：accuracy，精确度；latitude，纬度；longitude，经度；altitude，海拔；altitudeAccuracy，海拔高度的精确度；heading，朝向；speed，速度。

errorCallback 和 successCallback 函数一样，都带有一个参数，对象字面量格式，表示返回的错误代码。它包含以下两个属性：message，错误信息；code，错误代码。其中，错误代码包括以下 4 个值。

- unknow_error：表示不包括在其他错误代码中的错误，这里可以在 message 中查找错误信息；
- permission_denied：表示用户拒绝浏览器获取位置信息的请求；
- position unavalabl：表示网络不可用或者连接不到卫星；
- timeout：表示获取超时。必须在 options 中指定了 timeout 值时才有可能发生这种错误。

positionOptions 的数据格式为 JSON，有 3 个可选的属性：

- enableHighAcuracy：布尔值，表示是否启用高精度模式，如果启用这种模式，浏览器在获取位置信息时可能需要耗费更多的时间。
- Timeout：整数，表示浏览器需要在指定的时间内获取位置信息，否则触发 errorCallback。
- maximumAge：整数 / 常量，表示浏览器重新获取位置信息的时间间隔。

8.9 前端大“虾”必考题（编码问题）

第 368 道：若某 HTML 页面使用 GBK 字符集，CSS 文件使用 UTF-8 字符集，请简述如何可以避免乱码问题？

解析：

直接在 Web.config 里面设置 `<system.web><globalization fileEncoding="utf-8" requestEncoding="utf-8" responseEncoding="utf-8" culture="zh-CN"/></system.web>`。

第 369 道：如果一个网页上需要同时有多种语言文字，那么网页需要用_____编码格式。

答案：UTF-8

解析：

网页使用 UTF-8 编码唯一的好处是，无论你的操作系统的使用语言是简体中文（GB2312 字符集）、繁体中文（BIG5 字符集）或者是朝鲜文、日文、法文、德文、俄文、阿拉伯文、希伯来文、西班牙文、葡萄牙文等各种语言文字，在使用这些语言文字时，都可以正常显示在网页中，当其他任何人浏览时都会正常显示，不会有乱码，不会有重码和字符冲突，不需要调整页面的语言编码设置即可正常浏览，多种语言字符可以同时共存在页面上。UTF-8 是世界通用的语言编码，UTF-8 的推广要归功于 Google 的应用，以及 Blog 开发者。而如果用 Windows XP 英文版的 IE6 浏览 GB2312 语言编码的网页，则会提示是否安装语言包。因此，可能会失去很多的国外浏览者。UTF-8 是 Unicode（八位交换格式）的简称，Unicode 是国际标准，也是 ISO 标准 10646 的等价标准。Unicode 编码的文件中可以同时对几乎所有地球上已知的文字字符进行书写和表示，而且已经是 UNIX/Linux 世界的默认编码标准。在中国大陆，简体中文版常用的 GB2312/GB18030/GBK 系列标准是我国的国家标准，但只能对中文和多数西方文字进行编码。为了网站的通用性，用 UTF-8 编码是更好的选择。

综合提升

第 370 道：现在流行 HTML5+CSS3 和 jQuery 配合使用，请说出对其的理解，并举例说明以前做过的实例。

解析：

现在使用最流行的 Web 技术 HTML5+CSS3 和 jQuery，同样也可以实现类似的用户体验。使用这些特性将会比使用 Flash 更加有效。

以下示例仅供参考：

```
$(function () {
```



```
new Swipe(document.getElementById('jisou-banner'), {
    speed: 500,
    auto: 3000,
    callback: function () {
        var lis = $(this.element).next("ol").children();
        lis.removeClass("on").eq(this.index).addClass("on");
    }
});
```

第 371 道：你做的页面通常在哪些浏览器测试过？这些浏览器的内核分别是什么？经常遇到的兼容性问题有哪些？怎么会出现？解决方法是什么？移动终端主要考虑什么？

解析：

一般网页通过 IE6、IE7、IE8，Firefox5+，Chrome，Safari 测试。

- IE 内核浏览器：360，傲游，搜狗，世界之窗，腾讯 TT；
- 非 IE 内核浏览器：Firefox，Opera，Safari，Chrome。

关于经常遇到的兼容性问题，这个只需如实回答就好，排版中总会遇到兼容问题的，你只需要整理好问题，并把解决方法说出来。例如：

(1) IE6 双倍边距的问题。在使用了 float 的情况下，不管是向左还是向右都会出现双倍，最简单的解决方案就是把 display:inline; 加到 CSS 里面去。

(2) 文字本身的大小不兼容问题。同样是 font-size:14px 的宋体文字，在不同浏览器下占用的空间是不一样的。在 IE 下实际占高 16px，下留白 3px；在 Firefox 下实际占高 17px，上留白 1px，下留白 3px；在 Opera 下就更不一样了。解决方案：给文字设定 line-height。确保所有文字都有默认的 line-height 值。这点很重要，在高度上我们不能容忍 1px 的差异。

(3) 在 Firefox 下容器高度限定，即容器定义了 height 之后，容器边框的外形就确定了，不会被内容撑大，而在 IE 下是会被内容撑大的，高度限定失效。所以不要轻易给容器定义 height。

(4) 横向上的内容撑破容器问题。如果 float 容器未定义宽度，在 Firefox 下内容会尽可能撑开容器宽度，在 IE 下则会优先考虑内容折行。所以，内容可能撑破的浮动容器需要定义 width。

(5) 浮动的清除，在 Firefox 下必须清除浮动。

(6) mirrormargin Bug。当外层元素内有 float 元素时，外层元素如定义 margin-top:14px，将自动生成 margin-bottom:14px。padding 也会出现类似问题，都是在 IE6 下产生的，该类 Bug 出现的情况较为复杂，远不只这一种出现条件，还没系统整理。解决方案：外层元素设定 border 或设定 float。

(7) 吞吃现象。还是 IE6，上下两个 div，上面的 div 设置背景，却发现下

面没有设置背景的 div 也有了背景，这就是吞吃现象。对应上面的背景吞吃现象，还有滚动下边框缺失的现象。解决方案：使用 zoom:1。这个 zoom 是专门为解决 IE6 Bug 而生的。

第 372 道：简述从前端角度出发做好 SEO 需要考虑什么？

解析：

CSS Sprites：通俗来讲，就是图片合并，可以把网站中一些比较通用的小图片，合并到一张图片上，然后利用 CSS 技术来分别调用图片不同的部分。这样可以大大地减少 HTTP 的请求量，在网页加载时，速度就快很多，现在很多大中型网站都在用这个前端加速技术，效果也是很不错的。这也是很多人比较容易忽略的。在页面中，请你为每一个图片都指定一个 width 属性与 height 属性，这样在页面加载时，浏览器会预先留出既定的位置，图片下边的代码可以继续下载而不用等待，提高并行下载的速度，也提高了页面加载的速度。

启用 Keep-Alive 属性，Keep-Alive 可以理解为长连接，在没有启用 Keep-alive 属性之前，浏览器向服务器请求的 connection 是即连即断的，执行一次 HTTP 请求完成后，马上断开这个连接，而启用 connection 的 Keep-Alive 属性之后，这个连接可以保持一段时间，从而可以提高页面加载的速度。

使用浏览器缓存，可以用缓存技术来提高页面的加载速度，为一些不经常变化的文件，设置一个相对较长的过期时间，这样当用户访问网站后，就会在它的浏览器中留下缓存，当它在下次请求时，留在缓存中的组件就不用再向服务器发出 HTTP 请求了，这样减少了浏览器向网站服务器发出的 HTTP 请求数，从而提高了页面加载速度，这在一些图片比较多的网站上，效果是非常明显的，我们要善于使用缓存技术。

启用 GZIP 压缩，大中型网站，基本都启用了 GZIP 压缩，如果使用的是虚拟主机，可以让服务商来启用，如果有自己的服务器，启用也很简单。为什么启用 GZIP 压缩就会加快速度呢？因为当启用了 GZIP 后，网站服务器传输数据之前，是经过压缩了的，当传输到浏览器后，会再被解压缩，从而可以在浏览器上正常显示，而且压缩率可以达到很高，效果非常好。一般启用了压缩后，搜索引擎对网站的抓取量也是上升了的。

第 373 道：请简述如何通过 CSS 实现不同分辨率的手机适配，要求不可通过 width=100% 和留白的方式实现。

解析：

允许网页宽度自动调整。

首先，在网页代码的头部，加入一行 viewport 元标签。`<meta name="viewport" content="width=device-width, initial-scale=1" />`viewport 是网页默认的宽度和高度，上面这行代码的意思是，网页宽度默认等于屏幕宽度（width=device-width），原始缩放比例（initial-scale=1）为 1.0，即网页初始大小占屏幕面积的

100%。所有主流浏览器都支持这个设置，包括 IE9。对于那些老式浏览器（主要是 IE6、IE7、IE8），需要使用 CSS3-mediaqueries.js。

第 374 道：请根据以下 XML 写出对应的 JSON。

```
<xml>
  <list>
    <item>
      <id>12</id>
      <name>张三</name>
    </item>
    <item>
      <id>13</id>
      <name>李四</name>
    </item>
  </list>
</xml>
```

解析：

```
[[{id:12, name:'张三'}, {id:13, name:'李四'}]]
```

第 375 道：HTTP 状态码 301、302、401、404，分别代表什么意义？

解析：

301（永久移动），请求的网页已永久移动到新位置。服务器返回此响应（对 GET 或 HEAD 请求的响应）时，会自动将请求者转到新位置。应使用此代码告诉 Googlebot 某个网页或网站已永久移动到新位置。

302（临时移动），服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来响应以后的请求。此代码与响应 GET 和 HEAD 请求的 301 状态码类似，会自动将请求者转到不同的位置，但不应使用此代码来告诉 Googlebot 某个网页或网站已经移动，因为 Googlebot 会继续抓取原有位置并编制索引。

401（未授权），请求要求身份验证。对于登录后请求的网页，服务器可能返回此响应。

404（未找到），服务器找不到请求的网页。例如，对于服务器上不存在的网页经常会返回此代码。

第 376 道：如何优化页面速度，提高页面响应？

解析：

（1）如果 <head> 部分未定义字符集，那么将增加页面渲染次数，速度则会减慢。

（2）Meta 信息完善程度。建议网站 Meta 信息填写完整。

（3）合并域名。

（4）取消重定向。无论是通过服务器端重定向，还是 JavaScript 代码进行内容重定向，网站都会首先加载一个空白的页面，然后在定向到另外的页面，这

样不仅延长了页面的加载时间，还有可能导致无法跳转，让用户面对一个“空白”页面。

(5) 合并 JavaScript，清除多余的脚本。将网站中的 JavaScript 封装到一起，缩小 JavaScript 范围，比如去除不必要的空格等其他标签。

(6) 合并 CSS。将网站中的 CSS 封装到一起。

(7) 使用 CSS Sprite。CSS Sprite，在国内很多人称它为 CSS 精灵，是一种网页图片应用处理方式。

(8) 启用 GZIP。

(9) CSS 位置。建议将它们放到 <head> 标签中，页面需要重新渲染，打开速度受到影响。

(10) JavaScript 位置，JavaScript 放在页面最后，可以加快页面打开速度。

(11) 图片大小声明。如果图片大小不做定义，则页面需要重新渲染，速度受到影响。

(12) 图片 Alt 信息检测，建议为网页 img 标签增加 Alt 信息（检测信息为整个页面图片，加入这项信息可使网页上的图片更容易被用户检索到）。

(13) 去除错误链接，无法打开的链接，会导致页面打开缓慢，请及时修正或删除，并给网站增加 404 错误提醒页面。

(14) 缓存静态资源。变化很少的静态资源可以设置客户端缓存，这样可以减少请求次数。比如一篇文章发布后就不再修改了，这样就应该生成 HTML 文件，来提高网站加载速度。

(15) URL 长度检测，建议 URL 长度在 256byte 以内。

(16) 在静态页面上不建议使用动态参数。在静态页面上使用动态参数，会造成 spider 多次和重复抓取 URL。

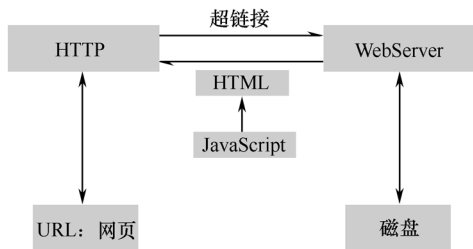
(17) 尽量不用 frame/iframe。

(18) Flash 勿出现不必要的文字信息。

(19) 减少域名 DNS 查找时间，用户在浏览器的地址栏中键入域名后，浏览器就会通过 DNS 系统查找域名对应的 IP 地址，所以需要将 DNS 的时间设置在较低的水平，比如平均 60 ~ 100ms 进行一次 DNS 查询。

第 377 道：使用 HTTP 协议、WebServer、HTML、URL、JavaScript 和 Web，描述从提交到返回数据呈现的过程。

解析：



第 378 道：如何提高前端性能？

解析：

1. 用 Web storage 替换 cookie

cookie 最大的问题是每次都会跟在请求后面。在 HTML5 中，用 sessionStorage 和 localStorage 把用户数据直接存在客户端，这样可以减少 HTTP 请求的数据量。

2. 使用 CSS 动画，而不是 JavaScript 动画

使用 CSS 的动画，而不是 JavaScript 动画。因为某些机器可以对 CSS 的动画进行 GPU 加速，而且也减少了 JavaScript 文件请求。

3. 使用客户端数据库

使用 Web SQLDatabase 或 IndexedDB 这类客户端数据库，可以减少 HTTP 请求的数量。像地区列表、好友列表这样的数据可以直接存储在客户端。有时也可以使用 sessionStorage 和 localStorage，因为一般来说，这两类相比 Web SQLDatabase 和 IndexedDB 更快。

4. 直接使用 JavaScript 的新功能

JavaScript 已经有了很大的发展，比如 Array 引入了很多新的方法，如 map、filter、forEach 等。另外，JSON 也直接嵌入浏览器了，不需要再引入 JSON2.js 文件了。

5. 缓存 HTML 标记

通过缓存，把 HTML 文件缓存在客户端。不过这些缓存的 HTML 文件只有结构，没有内容。内容需要通过 JavaScript 操作的 JSON 对象来把数据填入页面中这样的 HTML 文件相当于模板。

6. 使用硬件加速

现在领先的浏览器已经启用了 GPU 级别的硬件加速，通过某些指令或 hack 可以打开这些硬件加速。比如 CSS 中使用 3D 转换或动画，就可以打开 GPU 硬件加速。

第 379 道：当一个页面在你的电脑上没有问题，但在用户端有问题，你怎么办？

解析：

当页面在自己电脑上没问题而在用户端有问题时，我们应先检查自身是不是出错，按 F12 键打开浏览器，看控制台有没有报出错误，如果控制台没有报错信息，进入下一步，查看资源请求有没有出错，如果这些都没有问题，那么我们再想别的办法。换一台电脑，或换一个不同的网段再次进行测试，或者我们可以和用户联系，询问用户端的浏览器，检查是不是需要解决浏览器兼容问题。

第 380 道：请说出对于浏览器兼容性的看法，并列举一些处理方法。

解析：

浏览器兼容问题一：不同浏览器的标签，默认的 margin 和 padding 不同。

问题症状：随便写几个标签，在不加样式控制的情况下，各自的 margin 和 padding 差异较大。

碰到几率：100%。

解决方案：CSS 里加一行 `*{margin:0;padding:0;}`。

备注：这个是最常见的，也是最易解决的一个浏览器兼容性问题，几乎所有的 CSS 文件开头都会用通配符 * 来设置各个标签的内外补丁是 0。

浏览器兼容问题二：在块属性标签 float 后，有横行的 margin 情况下，在 IE6 下显示的 margin 比设置的大。

问题症状：常见症状是 IE6 中后面的一块被顶到下一行。

碰到几率：90%（稍微复杂点的页面都会碰到，是 float 布局最常见的浏览器兼容问题）。

解决方案：在 float 的标签样式控制中加入 `display:inline`；将其转化为行内属性。

备注：我们最常用的就是 DIV+CSS 布局了，而 DIV 就是一个典型的块属性标签，横向布局时我们通常都是用 DIV float 实现的，如果用 margin 设置横向的间距，这就是一个必然会碰到的兼容性问题。

浏览器兼容问题三：设置较小高度标签（一般小于 10px），在 IE6、IE7、遨游里，高度超出自己设置高度。

问题症状：在 IE6、IE7 和遨游里，这个标签的高度不受控制，超出自己设置的高度。

碰到几率：60%。

解决方案：给超出高度的标签设置 `overflow:hidden`；或者设置行高 `line-height` 小于你设置的高度。

备注：这种情况一般出现在我们设置小圆角背景的标签里。出现这个问题的原因是，在 IE8 之前的浏览器都会给标签设定一个最小默认的行高的高度。即使你的标签是空的，这个标签的高度还是会达到默认的行高。

浏览器兼容问题四：行内属性标签，设置 `display:block` 后采用 float 布局，又有横行的 margin 的情况，IE6 间距出现 bug。

问题症状：IE6 里的间距比超过设置的间距。

碰到几率：20%。

解决方案：在 `display:block`；后面加入 `display:inline; display:table`；。

备注：行内属性标签，为了设置宽高，我们需要设置 `display:block`；（除 input 标签比较特殊外）。在用 float 布局并有横向的 margin 后，在 IE6 下，它就具有了块属性 float 后的横向 margin 的 bug。不过因为它本身就是行内属性标签，所以我们再加上 `display:inline` 的话，它的高宽就不可设了。这时，我们还需要在 `display:inline` 后面加入 `display:table`。

浏览器兼容问题五：图片默认有间距。

问题症状：几个 `img` 标签放在一起时，有些浏览器会有默认的间距，即使加了问题一中提到的通配符也不起作用。

碰到几率：20%。

解决方案：使用 `float` 属性为 `img` 布局。

备注：因为 `img` 标签是行内属性标签，所以只要不超出容器宽度，`img` 标签都会排在一行里，但是部分浏览器的 `img` 标签之间会有个间距。去掉这个间距，使用 `float` 是正道。（我的一个学生使用负 `margin`，虽然能解决，但负 `margin` 本身就是容易引起浏览器兼容问题的用法，所以我禁止他们使用）。

浏览器兼容问题六：标签最低高度设置 `min-height` 不兼容。

问题症状：因为 `min-height` 本身就是一个不兼容的 CSS 属性，所以在设置 `min-height` 时，不能很好地被各个浏览器兼容。

碰到几率：5%。

解决方案：如果我们要设置一个标签的最小高度 200px，需要进行的设置为：`{min-height:200px; height:auto !important; height:200px; overflow:visible;}`。

备注：在 B/S 系统前端开发时，在很多情况下我们有这种需求。当内容小于一个值（如 300px）时，容器的高度为 300px；当内容高度大于这个值时，容器高度被撑高，而不是出现滚动条。这时我们就会面临这个兼容性问题。

浏览器兼容问题七：各种特殊样式的兼容，比如透明度、圆角、阴影等。特殊样式在每个浏览器中的代码区别很大，所以，只能现查资料，通过给不同浏览器写不同的代码来解决。

JavaScript 解决 IE6 下 png 透明失效的问题。

做兼容页面的方法是：每写一小段代码（布局中的一行或者一块），我们都要在不同的浏览器中看是否兼容，当然熟练到一定的程度就没这么麻烦了。建议经常碰到兼容性问题的新手使用。很多兼容性问题都是因为浏览器对标签的默认属性解析不同造成的，只要我们稍加设置都能轻松地解决这些兼容问题。如果我们熟悉标签的默认属性的话，就能很好地理解为什么会出现兼容问题及怎么去解决这些兼容问题了。

第3篇

箭在弦

——DIV+CSS 向前冲

，
不回头

DIV+CSS 是 Web 设计标准，它是一种网页的布局方法，与传统的通过表格（table）布局定位的方式不同，它可以实现网页页面内容与表现相分离，而且它结构清晰，很容易被搜索引擎搜索到，天生就适合 SEO 优化，降低网页大小，让网页体积变得更小。

第9章



夯实基础，厚积薄发 [DIV+CSS初级面试题]

登上理想的高峰——块级元素，CSS 概述不曾退缩，float:left、right、center 让你腾云驾雾，img 图片让你的背景更加风采，绝地逢生之路——CSS 规范。

9.1 有一种题叫边做边流泪（浮动）

第 381 道：如何清除一个标签的子标签的浮动？

解析：

1. 空标签清除浮动 `.clear{clear: both;}`

优点：通俗易懂，容易掌握。

缺点：会添加大量无语义空标签，结构与表现未分离，不利于维护。

2. br 标签清除浮动

优点：比空标签方式语义稍强，代码量较少。

缺点：结构与表现未分离，不推荐。

3. 父元素设置 `overflow: hidden`

优点：不存在结构和语义化问题，代码量少。

缺点：内容增多时容易造成不会自动换行的后果，导致内容被隐藏，无法显示需要溢出的元素。IE6 需要触发 haslayout。

4. 父元素设置 `overflow: auto`

优点：不存在结构和语义化问题，代码量少。

缺点：多个嵌套后，Firefox 在某些情况下会造成内容全选；IE 在 mouseover 造成宽度改变时会造成最外层模块出现滚动条。IE6 需要触发 haslayout。

5. 父元素浮动

优点：不存在结构和语义化问题，代码量少。

缺点：与父元素相邻的元素的布局会受到影响，不可能一直浮动到 body，不推荐。

6. 父元素设置 `display: table`;

优点：结构语义化完全正确，代码量少。

缺点：盒模型属性已经改变，不推荐。

7. 使用 :after 伪元素（不是伪类）

优点：结构和语义化完全正确，代码量居中。

缺点：复用方式不当会造成代码量增加。由于 IE6 和 IE7 不支持 :after，使用 zoom:1 触发 haslayout。

第 382 道：描述一下浮动会造成什么影响，如何居中一个浮动元素？

解析：

我们知道，当一个元素浮动时，其周围的文字段落等会相应地环绕它。这样浮动的元素就会成为一座“孤岛”，周围的东西就会成为“水流”围绕着它。这时问题就来了，我们往往希望浮动不要对周围原有的布局产生影响想到的是利用 clear 来清除浮动的影响，但是却忘记了 clear 只对块级元素有用，所以经常会因为忽略这点导致清除影响失败。

居中一个元素：首先设置块元素的宽度和高度，这里宽度必须得设置，高度可以不设置。设置块的背景色，实现居中的关键在于 margin-left:50%; position:relative;left:-250px;

设置 margin-left:50%; 后，浮动元素左边正好位于文档中间，设置块相对定位 position:relative; 然后左移宽度的二分之一，即可实现元素居中。在这个实例中宽度设置为 500px，left 设置为 -250px。

第 383 道：简单说明绝对定位和浮动的区别和应用。

解析：

当一个元素使用绝对定位后，它的位置将依据浏览器左上角开始计算或相对于父容器（在父容器使用相对定位时）。绝对定位使元素脱离文档流，因此不占据空间。普通文档流中元素的布局就与绝对定位的元素不存在时一样，因为绝对定位的框与文档流无关，所以它们可以覆盖页面上的其他元素。

而浮动元素的定位还是基于正常的文档流，然后从文档流中抽出并尽可能远的移动至左侧或者右侧，文字内容会围绕在浮动元素周围。当一个元素从正常文档流中抽出后，仍然在文档流中的其他元素将忽略该元素并填补原先的空间。它只是改变了文档流的显示，而没有脱离文档流，一个元素浮动或绝对定位后，它将自动转换为块级元素，而不论该元素本身是什么类型。

第 384 道：当 float 和 margin 同时使用时，如何解决 IE6 的双倍边距 Bug？

解析：

解决办法就是用 display:inline;

display:inline 的作用是设置对象作为行内元素显示，inline 是内联对象的默认值（内联对象就是不自动产生换行的元素，比如 span），而我们一般用的 DIV 是块级元素，默认 display 属性是 block，但将 DIV 的 display 设置为 inline 的话，则多个 DIV 可以像 span 一样显示在一行了。

第 385 道：当内部容器使用了 float: left 后，你会用哪些方法对父元素消除

浮动？

解析：

- (1) 给父级 DIV 也加上 float。
- (2) 在子级 DIV 后面加上 `<div style="clear:both;"></div>`，即清除浮动。
- (3) 给父元素加上 `overflow:hidden;` 属性。

9.2 想登上理想的高峰吗？那就来吧（块级元素）

第 386 道：块级元素和内联元素的区别？

解析：

块级元素独占一行，在默认情况下，宽度自动填满父元素宽度。

内联元素不会独占一行，宽度随着元素的内容而变化，并且 `width` 和 `height` 的设置无效，同时在 `margin` 和 `padding` 的垂直方向上不产生边距效果，可以使用 `display:block;` 或 `display:inline;` 实现块级元素和行内元素的相互转换。

第 387 道：`display` 属性值的常用取值不包括（ ）

- A. inline B. block C. hidden D. none

答案：C

解析：

- `display:inline` 通常被叫作行内元素。
- `display:block` 通常被叫作块级元素。
- `display:none` 在不删除元素的情况下通常被 JavaScript 用来隐藏或显示元素。

9.3 时间很短，面试赶紧（行内元素）

第 388 道：行内元素有哪些？块级元素有哪些？

解析：

行内元素：`span` 为常用行内元素，定义文本内区块。行内元素还有：`a` 为锚点，`abbr` 为缩写，`acronym` 为首字，`b` 为粗体，`big` 为大字体，`br` 为换行，`cite` 为引用，`em` 为强调，`dfn` 为定义字段，`font` 为字体设定，`i` 为斜体，`img` 为图片，`input` 为输入框，`kbd` 为定义键盘文本，`label` 为表格标签，`q` 为短引用，`s` 为中画线，`samp` 为定义范例计算机代码，`select` 为项目选择，`small` 为小字体文本。

块级元素：`address` 为地址，`div` 为常用块级元素，也是 `CSSlayout` 的主要标签。`dl` 为定义列表，`form` 为交互表单，`h1` 为大标题，`h2` 为副标题，`h3` 为 3 级标题，

h4 为 4 级标题, h5 为 5 级标题, h6 为 6 级标题, hr 水平分隔线, menu 为菜单列表, ol 为排序表单, p 为段落, pre 为格式化文本, table 为表格, ul 为非排序列表。

第 389 道：简述 div 元素和 span 元素的区别。

解析：

div (division) 是一个块级元素，可以包含段落、标题、表格，比如章节、摘要和备注等。而 span 是行内元素，span 的前后是不会换行的，它没有结构的意义，纯粹是应用样式，当其他行内元素都不合适时，可以使用 span。块级元素和行内元素也不是一成不变的，只要给块级元素定义 display:inline，块级元素就成了内嵌元素，同样地，给内嵌元素定义 display:block 就成了块级元素。

9.4 面试就像半杯水，你能看到什么（CSS 图片）

第 390 道：如何让 img 标签在 DIV 里左右、上下居中？

解析：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD Xhtml 1.0 Transitional
//EN" "">
<html xmlns="">
<head>
<meta name="author" type="Nangle from CAU CS101" />
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
<title> 如何让 img 标签在 DIV 里左右、上下居中 </title>
<script type="text/JavaScript"> </script>
<style type="text/css">
#container{width:400px;height:400px;background:#ccc;
text-align:center;}
img{margin:100px auto 0 auto;}
</style>
</head>
<body>
<div id="container">
<img src="" width="200" height="200" />
</div>
<p>
</p>
</body>
</html>
```

9.5 面试是一场旅行，正在进行时（CSS 引入）

第 391 道：CSS 引入的方法有哪些？都有什么区别？

解析：

(1) 在 head 部分加入 `<link rel="stylesheet" type="text/css" href="my.css"/>`，引入外部的 CSS 文件。

这种方法是目前引入最多的方法，它不仅最能体现 CSS 的特点，而且也最能体现 DIV+CSS 中的内容与显示分离的思想，改版维护容易，代码美观。

(2) 在 head 部分加入 `<style text="text/css"></style>`

这种方法的使用情况要少得多，最常见的就是访问量大的门户网站，或者访问量较大的企业网站的首页。与第一种方法比起来，这种方法优点突出，弊端也明显。

- 优点：速度快，所有的 CSS 控制都是针对本页面标签的，没有多余的 CSS 命令；再者不用外链 CSS 文件，直接在 HTML 文档中读取样式。
- 缺点：改版麻烦，单个页面显得臃肿，CSS 不能被其他 HTML 引用，造成代码量相对较多，维护也麻烦。对它们来说，用户量是关键，它们需要大量人员进行复杂的维护工作。

(3) 直接在页面的标签里加 `<div style="border:1px red solid;"> 测试信息 </div>`。

这种方法现在用得很少，一般情况下，我们认为 HTML 里不能出现 CSS 命令，其实有时候使用一下也未尝不可。比如通用性差，效果特殊，使用 CSS 命令较少，并且不常改动的地方，使用这种方法反而是很好的选择。

(4) `@import url(my.css);` 这种方法引入 CSS 文件。

第 392 道：引入样式表的方式有几种？（ ）多选

A. Include B. Import C. link D. Ref

答案：ABC

解析：

```
@import url(my.css);  
<link rel="stylesheet" type="text/css" href="my.css"/>
```

9.6 面对考题不曾退缩（CSS 概述）

第 393 道：什么是 CSS？

解析：

级联样式表（Cascading Style Sheet），简称“CSS”，通常又称为风格样式表（Style Sheet），它是用来进行网页风格设计的。比如，如果想让链接字未单

击时是蓝色的，当鼠标移上去后字变成红色的且有下画线，这就是一种风格。通过设立样式表，可以统一地控制 HTML 中各标志的显示属性。级联样式表可以使人更有效地控制网页外观。使用级联样式表，可以精确指定网页元素位置、外观，并且具有创建特殊效果的能力。

第 394 道：在书写高效 CSS 文件时，需要考虑哪些问题？

解析：

- (1) 不影响页面的布局。
- (2) 去掉不必要的样式。
- (3) 合并相同的样式。
- (4) 尽可能缩小样式的大小。

第 395 道：CSS 的基本语句构成是什么？

解析：

选择器 { 属性 1: 值 1; 属性 2: 值 2;……}

有 6 种合法的选择器类别，即：HTML 标记、具有上下文关系的 HTML 标记、用户定义的类、用户定义的 ID、虚类和虚元素。

1. HTML 标记

例如：`h1 {text-align:center;font-family: 楷体 -gb2312}`
`h1,h2,h3 {color:#ff0000}`

2. 具有上下文关系的 HTML 标记

例如：`h1 B {color:red}`

这表示只有位于 <h1> 标记元素内的 标记符说明的内容显示为红色，其他 标记符说明的元素不具有该属性。这种上下文关系可以多层嵌套。

3. 用户定义的类

例如：要想将一个类包括到样式定义中，可将一个句点和一个类名添加到选择器后，即：选择器.类名 { 属性: 值;……}

4. 用户定义的 ID

例如：ID，就相当于 HTML 文档中样式的“身份证”，以保证其在一个 HTML 文档中具有唯一可用的值。这给使用 JavaScript 等脚本编写语言的应用带来了方便。要将一个 ID 包括在样式定义中，需要“#”号作为 ID 名称的前缀，格式如下：`#ID 名字 { 属性: 值 }`

5. 虚类和虚元素

例如，显示方式：

```
A:link {text-decoration:none;color:#800000}
A:visited {text-decoration:none;color:#800000}
A:active {text-decortion:none}
A:hover {text-decoration:underline;color:blue}
```

第 396 道：简述一下什么是内容与表现分离？CSS 都有哪些调用方式？

解析：

代码中不涉及任何的表现元素，如 `style`、`font`、`bgColor`、`border` 等，也就是说，内容编写在 HTML 中，而样式编写在 CSS 中。

CSS 调用方式：

1. 使用 `link` 标记

```
<link rel="stylesheet" type="text/css" href="sheet.css" />
```

2. 使用 `style` 元素

```
<style type="text/css">
    body{background:#fff;}
    h1{font-size:12px;}
</style>
```

3. 使用 `@import` 指令

```
<style type="text/css">
    @import url(sheet1.css);
    @import "sheet2.css";
</style>
```

4. 使用 `style` 属性的内联样式（inline style）

```
<p style="color:#f00;">这是红色的字</p>
```

第 397 道：CSS 层叠是什么？

解析：

CSS 层叠指的是样式的优先级，当产生冲突时以优先级高的为准。

(1) 开发者样式 > 读者样式 > 浏览器样式（除非使用 `!important` 标记）。

(2) id 选择符 >（伪）类选择符 > 元素选择符。

(3) 权重相同时取后面定义的样式。

以下是一段经典的 HTML，三个类名分别为模块、标题和正文。

```
<div class="mod">
    <div class="hd"></div>
    <div class="bd"></div>
</div>
```

大部分 HTML 页面都可以由这种结构嵌套或者累加而成。

9.7 面试者就像蒲公英，看似自由，却身不由己（CSS 选择器）

第 398 道：请解释浏览器是如何根据 CSS 选择器选择对应元素的？

解析：

1. 子选择器（>）

在 IE7+ 标准模式，以及 Chrome，Firefox 下开始支持。

2. 临近兄弟选择器 (+)

在 IE7+ 标准模式，以及 Chrome，Firefox 下开始支持。但是，如果父元素与子元素之间有注释，那么就会失效。IE8 没有。

3. 普通兄弟选择器 (~)

选择该元素后面的所有兄弟节点，在 IE7+ 标准模式，以及 Chrome，Firefox 下支持。它和临近兄弟节点选择器的区别就是，前者就选择后面所有的，不求相邻，但是后面必须相邻，且选择一个。

第 399 道：以下有若干个 CSS 选择器，请给出它们的优先级顺序。

div h1、#div h1、h1、div h1 #_h、div h1.c_h。

解析：

div h1 #_h>#div h1>div h1.c_h>div h1>h1

第 400 道：CSS 选择器有哪些？

解析：

- 常见选择器：通配选择器（写法为 *{ 属性：值 }）；元素选择器、类选择器、id 选择器；
- 子集选择器：#nav>li{ 属性：值 }，注意：只针对 #nav 下的儿子元素 li；
- 同胞选择器：h1+p{ 属性：值 }，注意：紧跟着 h1 的 p 元素继承；
- 关联选择器：两个以上选择器组合，优先级等于权重之和；
- 伪选择器：::before 相当于在 HTML 上增加一个 DOM 节点;p::first-letter{font_size:2em}；
- 属性选择器：input[name]&&input[title="email"]&&input[条件 1][条件 2]{ 属性：值 }。

9.8 绝地逢生之“路”（CSS 规范）

第 401 道：CSS 规范中，每一条样式的结束是：“（分号）”还是：“（冒号）”？样式名与样式值之间的分隔符号是：“（冒号）”还是：“（分号）”？

解析：

CSS 规范中，每一条样式的结束是：（分号）；样式名与样式值之间的分隔符号是：（冒号）。

第 402 道：CSS 规范中，句号后面跟一个名称代表什么含义？

解析：

句号后面跟一个名称代表文档中所有 class 属性值包含这个名称的应用样式。

9.9 能磨炼薄弱意志的考题（HTML 结构）

第 403 道：在 HTML 代码中如何做 SEO？

解析：

（1）h 标签的使用，值得注意的是，不论任何页面，h1 标签只能出现一次，它是当前页面的主标题，权重最高，对蜘蛛的吸引力是最强的。

（2）strong 标签的使用，strong 标签对关键词的强调作用仅次于 h 标签，用于加粗段落标题或是重点关键词。

（3）<title> 网站 SEO 标题 </title>、<meta name="description" content=" 网站描述 "> 和 <meta name="keywords" content=" 网站关键词 ">，这是 SEO 的重点。

（4） 链接关键词 ，站内丰富的超链接会方便蜘蛛爬行，体现网站的深度和广度，这点在 SEO 中至关重要。

（5），这是针对网页中图片的，当然也可以写成 。

（6）<div id="copyright"> 版权部分加上网站名称和链接 </div>。

（7）HTML 优化要富于逻辑，重点明确，层次分明，这也是符合 SEO 精神的。

第 404 道：<div style="font-size:12px;width:20em;">www.yinyuetai.com</div> 说出上列代码中 div 的实际高度和宽度，以及 span 对象的实际宽度和高度。

解析：

div 的高度是 30px，宽度是 240px；span 的宽度是 60px，高度是 185px。

第 405 道：CSS 怎样判断不同分辨率显示不同宽度布局，从而实现自适应宽度？

解析：

在 CSS DIV 网页布局中，当分辨率小于等于 1024px（像素）时，DIV 布局对象显示 1000px 宽度；当分辨率大于 1024px 时，DIV 布局对象显示 1200px 宽度等需求。使用 CSS 实现改变浏览器显示宽度，从而实现布局的网页宽度动态变化（网页宽度自动随浏览器显示宽度而变宽变窄）。

随着科技的发展，电脑用户显示屏分辨率越来越高，但有的用户还是使用 1024px 的分辨率的显示屏（根据几个浏览器分辨率统计平台得到数据，现在使用 1200px 分辨率以下的用户极少，我们在 CSS 布局时至少需要考虑 1024px 分辨率的用户），如果网页布局的宽度固定到 1200px，当 1024px 分辨率用户浏览网页时，浏览器下方会出现滚动条，为了解决这个问题，大家可以通过使用 CSS3 样式判断用户浏览器宽度，从而调用不同布局宽度。

第 406 道：一个页面（HTML）有哪几部分构成，分别是什么？作用是什么？

解析:

HTML 由网页头和主体两部分组成, 分别是 `<head></head>` 和 `<body></body>`。

head 提供一些由浏览器等软件在解释转换 HTML 文档身体部分数据之前所需的相关信息, 这些信息一般都不会在浏览器窗口内显示, 但会影响到文档的解释转换与输出。文档头部信息放在 `<head>...</head>` 之间, 至少应在文档头部用 `<title>` 标记为文档设置标题信息, 文档标题信息显示在浏览器的标题栏中。

文档的身体部分放在 `<body>...</body>` (普通网页文档) 之间, 普通网页文档的身体部分包含了该文档需要输出的数据, 包括内容数据和控制内容输出的标记。浏览器等软件在解释转换该文档时, 会把文档身体部分的所有数据解释转换并用浏览器窗口显示等方式输出。

第 407 道: 你如何理解 HTML 的语义化? 是否接触或了解重构?

解析:

根据内容的结构化 (内容语义化), 选择合适的标签 (代码语义化) 不仅便于开发者阅读和写出更优雅的代码, 而且能让浏览器的爬虫和机器很好地解析。

当系统发展到一定阶段后, 使用重构的方式, 不改变系统的外部功能, 只对内部的结构进行重新的整理。通过重构, 不断地调整系统的结构, 使系统对于需求的变更始终具有较强的适应能力。

第 408 道: 下面代码片段, 说法正确的是 ()。

```
div1{position:absolute;
line-height:22px;
height:58px;
background-color:#ff0000;
}
```

- A. line-height:22px; 修饰文字字体大小
- B. position:absolute; 表示绝对定位, 被定位的元素位置固定
- C. height:58px; 表示被修饰的元素与别的元素的距离
- D. background-color:#ff0000; 表示被修饰的元素的背景图像

答案: B

解析:

- A. 修饰文字的行高。
- B. 表示绝对定位, 被定位的元素位置固定。
- C. 表示被修饰的元素的高度。
- D. 表示被修饰的元素的背景颜色。

position:absolute; 表示绝对定位, 被定位的元素位置固定。

第 409 道: HTML 代码 `<select name="name"></select>` 表示 ()。

- A. 创建表格
- B. 创建一个滚动菜单
- C. 设置每个表单项的内容
- D. 创建一个下拉菜单

答案: D

解析:

<select> 就是创建一个下拉菜单。

第 410 道: 在 HTML 文档中, 引用外部样式表的正确位置是 ()。

A. 文档的末尾 B. 文档的顶部 C. <body> 部分 D. <head> 部分

答案: D

解析:

引用外部样式一般是放在 <head></head> 中。

第 411 道: HTML 是 () 大小写的语言。

A. 区分 B. 不区分

答案: B

解析:

HTML 不区分大小写, 它只是标记性语言, 不是脚本 script。

9.10 看的越少, 失去的越多 (隐藏 DOM 元素)

第 412 道: 如何区别 CSS 中的 display:none 与 visibility:hidden ?

解析:

- display:none 视为不存在, 且不加载。

`CSS display:none;`

使用该属性后, HTML 元素 (对象) 的宽度、高度等各种属性值都将 “丢失”;

- visibility:hidden 表示隐藏, 但在浏览时保留位置。

`visibility:hidden;`

使用该属性后, HTML 元素 (对象) 仅仅是在视觉上看不见 (完全透明), 而它所占据的空间位置仍然存在, 也就是说它仍具有高度、宽度等属性值。

第 413 道: CSS 中能够隐藏 DOM 元素的属性都有哪些? 它们的区别是什么?

解析:

CSS 隐藏 DIV 可以写为 display:none; 或者 visibility:hidden;。不过这里 display:none; 是不占用空间的, 而 visibility:hidden; 虽然隐藏了, 但还占据着网页的空间。position 的 absolute (绝对定位) 生成绝对定位的元素, 相对于 static 定位以外的第一个父元素进行定位。元素的位置通过 “left”、“top”、“right” 以及 “bottom” 属性进行规定。position 的 relative (相对定位) 生成相对定位的元素, 相对于其正常位置进行定位。因此, “left:20” 会向元素的 left 位置添加 20 px。

第 414 道: 如何显示 / 隐藏一个 DOM 元素?

解析:

Show: 显示元素; hide: 隐藏元素。

9.11 喜欢前进，看的题就越来越多（CSS 文字样式）

第 415 道: color: #666666; 可缩写为 _____。

解析:

(color:#666)。

第 416 道: 给文字添加阴影用哪个属性?

解析:

text-shadow()。

第 10 章



百折不挠，历经磨难 [DIV+CSS中级面试题]

打开面试的钥匙——CSS 与表单，CSS 定位是王牌，DIV 布局囤积自信，display 教你如何隐藏，躲避敌人的追踪，让你若隐若现。

10.1 有一种题做起来很崩溃（inline-block 特性）

第 417 道：display 属性值的常用取值不包括（ ）。

A. inline B. block C. hidden D. none

答案：C

解析：

hidden 是用于隐藏的。

10.2 快到题里来（布局）

第 418 道：实现左侧宽为 200px，右侧自适应宽度的布局。

解析：

```
<div style="width:200px;height:500px;position:absolute;left:0;
top:0;background:#999999;"></div>
<div style="margin-left:200px;height:100%;background:#CCCCC;
height:100px;font-size:14px;color:#000;"> 实现左侧规定宽 200，右
侧自适应宽度的布局 </div>
```

第 419 道：在空白处加上 CSS，让这个 DIV 在页面中居中（上下左右都居中）。

```
<div style="border:1px solid black"; width:200px;
height:200px;>hello</div>
```

解析：

```
position:absolute;
top:50%;
left:50%;
margin-top:-100px;
margin-left:-100px;
```

第 420 道：创建一个文本框，让其宽为 120px，高为 20px，对齐方式为居中对齐，写出代码。

解析：

```
<input type="text" style="width: 120px; height: 20px; text-align: center"/>
```

10.3 思想太满，就会学不来（CSS 属性）

第 421 道：列举不重复的 10 个 CSS 属性。

解析：

- font-family：规定文本的字体系列；
- margin-top：设置元素的上外边距；
- margin-right：设置元素的右外边距；
- margin-bottom：设置元素的下外边距；
- margin-left：设置元素的左外边距；
- border-top-width：设置上边框的宽度；
- font-style：规定文本的字体样式；
- font-variant：规定是否以小型大写字母的字体显示文本；
- font-weight：规定字体的粗细；
- font-size：规定文本的字体尺寸。

第 422 道：标签上 alt 与 title 属性的区别是什么？

解析：

alt 是给搜索引擎识别时，在图像无法显示时的替代文本；title 是关于元素的注释信息，主要是给用户解读的。当鼠标放到文字或是图片上时，有 title 文字显示。

第 423 道：如何给元素添加圆角属性？

解析：

border-radius

10.4 面试失败十次，找第十一次坚持的借口（清除浮动与闭合浮动）

第 424 道：清除浮动和闭合浮动的不同点？

解析：

清除浮动：清除对应的单词是 clear，对应 CSS 中的属性有 4 个值分别是，

是 clear、left、right、both、none。

闭合浮动：更确切的含义是使浮动元素闭合，从而减少浮动带来的影响。

第 425 道：写出 CSS 清除元素浮动常用的方法。

解析：

(1) 使用空标签清除浮动，即 clear: both;。

(2) 使用 overflow 属性，即 overflow:auto。“zoom:1”用于兼容 IE6。

10.5 多项选择，任你选择（CSS 定义标签）

第 426 道：em 为什么可以缩放，以什么为基准？

解析：

一个 em 表示一种特殊字体的大写字母 M 的高度。在网页上，一个 em 是网页浏览器的基础文本尺寸的高度，一般情况下它是 16px。然而，任何人都可以改变这个基础尺寸的设置，因此，一个 em 对于有的人来说可能是 16px，但是在其他人的浏览器上可能是 24px。换句话说，em 是一个相对的度量单位。

第 427 道：下列标签可以不成对出现的是（ ）

A. <html></html> B. <p></p> C. <tirle></title> D. <body></body>

答案：B

解析：

<html> 与 </html> 标签限定了文档的开始点和结束点，在它们之间是文档的头部和主体，<title> 标签是 <head> 标签中唯一要求包含的东西，是网页设计里面的一对标识，表示网页主体，在它之间的东西是可以在网页展示的。这是网页设计中定义段落的标记，<p> 称为开始标记，</p> 称为结束标记。

第 428 道：id 为 left 的 DIV 标签，用 CSS 设置 DIV 的左边为红色实线，下面设置正确的是（ ）

A. style="border-top:#ff0000 1 solid;"

B. style="border-left:1, #ff0000, solid;"

C. style="border-left:1 #ff0000 solid;"

D. style="border-left:1, #ff0000, dashed;"

答案：C

解析：

第三个选项设置边框线为左，颜色为红色实线。

第 429 道：CSS 中的 id 标签和 class 标签，哪个定义的级别高？

解析：

id 的级别高于 class。

第 430 道：在空白处填代码：

```
<div onclick=" ">单击这里，让字体颜色变红，背景变蓝 </div>
```

解析：

```
color: red;background: blue;
```

第 431 道：在使用下面的标题符号时，哪一个字体是最大的（ ）

A. <h1> B. <h2> C. <h3> D. <h4>

答案：A

解析：

<h1> 标签是字体最大的，<h4> 标签是字体最小的。

第 432 道：使文字加粗的除了 之外，还有 _____。

解析：

strong 也是给文字加粗的。

第 433 道：下面哪一个标签可以产生一个回车换行（ ）

A. <hr> B.
 C. <tr> D. <tl>

答案：B

解析：

<hr> 表示一条水平线，
 表示插入一个简单的换行符，<tr> 表示在 HTML 表格中的一行，* tr 标签是成对出现的，以 <tr> 开始，</tr> 结束。

第 434 道：下面哪一个是符号 & 的符号码（ ）

A. " B. & C. &comp D. nbsp;

答案：D

解析：

 nbsp; 是 & 的符号码。

第 435 道：font-size: 62.5%，解释一下如此设计字体大小的原因。

解析：

这主要是为了方便 em 和 px 的相互转换，em 的初始值为 1em=16px，这样的话，1.2em=19.2px，可是我们在设置时很少看见 19.2px 这样表示的大小，也就是说，在用 px 表示大小时，数值是不带小数位的。

10.6 不满足昨天的难度（简化 CSS 代码）

第 436 道：请简化下面的 CSS 代码：

```
a) margin: 0px;
b) Padding: 10px 0 10px 0;
c) Border-width: 1px; border-style: solid; border-color:
#ff5500;
```


解析：

图片的链接，图片名字为 name，且 align="left" 表示图像相对于周围的文本左对齐。

10.8 缘分是一本书，翻得不经意会错过（CSS 设置表格）

第 441 道：HTML 语言中，设置表格边框宽度的标签是（ ）

- A. <table border=#> B. <table cellpadding=#>
C. <table cellspacing=#> D. <table width=#%>

答案：A

解析：

<table border=""> 里面可以添加很多布局元素，B 是设置表格外间距的；C 是设置表格内间距；D 是设置表格的宽度而非边框宽。

第 442 道：下面哪一个是表格单元格标记（ ）

- A. <tr> B. <tl> C. <td> D. <tp>

答案：C

解析：

<table> 定义一个表格，每一个表格只有一对 <table></table>，一张页面中可以有多个表格；<tr> 定义表格的行，一个表格可以有多个行；<td> 定义表格的单元格，每行可以有不同数量的单元格。

第 443 道：判断题：不只表格可以用来对页面进行布局，层也同样可以。

解析：

对，表格和层都可以对页面进行布局，现在用层的比较多。

第 444 道：在 table 中，tr 是 _____，td 是 _____。

答案：行，列

解析：

<tr> 定义表格的行，一个表格可以有多个行，<td> 定义表格的单元格，每行可以有不同数量的单元格。

第 445 道：HTML 中，定义表格的宽度时，80px 与 80% 的区别是什么？

解析：

px 表示像素，% 表示占整个页面的百分比。

10.9 不要在错的题上犹豫不决（背景图片）

第 446 道：以下 DIV 的背景是什么颜色的？

```
.a{
    background-color:#FF0000;
}
.b{
    background-color:#00FF00;
}
<div class="b a"></div>
```

解析:

#00FF00

b 覆盖了 a。

第 447 道: CSS3 中背景图新增的属性是什么? 属性值有哪些?

解析:

新增的属性:

- (1) background-clip 指定背景的显示范围。
- (2) background-origin 指定绘制背景图像的起点。
- (3) background-size 指定背景图片的尺寸大小。
- (4) background-break 指定内联元素的背景图像进行平铺时的循环方式。

属性值:

- (1) border-box 从边框区域向外裁剪背景。
- (2) padding-box 从补白区间向外裁剪背景。
- (3) content-box 从内容区域向外裁剪背景。
- (4) no-clip 从边框区域向外裁减背景。

10.10 面试是一把锁，你拿对钥匙了吗（CSS 与表单）

第 448 道: 对于 <from action="URL" method=*> 标签，其中 * 代表 GET 或 ()

- A. SET B. PUT C. POST D. INPUT

答案: C

解析:

表单数据可以作为 URL 变量 (method="get") 或者 HTTP post(method="post") 的方式来发送。

第 449 道: 在 Form 中，method 属性的哪种方法更具有保密性?

- A. GET B. POST

答案: B

解析:

如果希望获得最佳表单传输性能，可以采用 GET 方法发送只有少数简短字

段的小表单。如果安全性是个问题，那么我们建议选用 POST 方法。

第 450 道：在 Form 中的 input 可以设置 readonly 和 disable，请问这两项属性有什么区别？

解析：

readonly 只针对 input (text/password) 和 textarea 有效，而 disabled 对所有的表单元素都有效，包括 select、radio、checkbox、button 等。

第 451 道：关于表格表述正确的有 () (多选)

- A. 表格中可以包含 tbody 元素
- B. 表格中可以包含 caption 元素
- C. 表格中可以包含多个 tbody 元素
- D. 表格中可以包含 colgroup 元素
- E. 表格中可以包含 col 元素

答案：ABCDE

解析：

表格结构用到的标签有 caption、col、colgroup、tbody、td、tfoot、th、thead、tr，其中一个表格可以包含多个 tbody，但只能有一个 thead 和一个 tfoot。

第 452 道：在 Form 中，input 有哪些类型？各是做什么处理使用的？

解析：

在 Form 中，input 是拥有类型最多的元素，分别有：text、password、radio、checkbox、file、button、image、submit、reset、hidden 等。

text：文本框；password：密码框；radio：单选按钮；checkbox：复选框；file：文件选择框；button：单纯的按钮；image：后面必须跟随 src 属性并指定一张图片，它的作用是用一张图片作为按钮；submit：提交按钮；reset：重置按钮，重置表单中的所有内容，可以让用户重新写入；hidden：一个隐藏元素，用于提交表单时附带一些用户不需要填写的默认值，比如附件的容量、大小等。

10.11 有思维才是王牌 (CSS 定位)

第 453 道：编写 CSS 让 div2 在 div1 的右下角。

解析：

```
#div1{position:relative;}, #div2{right:0px; bottom:0px;
position:absolute;}
```

第 454 道：请列举 CSS 中 position 的参数和作用。

解析：

position 有 4 个可选值，分别是：static、absolute、fixed、relative。

`position: static`——无定位，改属性值是所有元素定位的默认情况，在一般情况下，我们不需要特别去声明，但在遇到继承的情况时，我们不愿意见到元素所继承的属性影响本身，从而可以用改属性来取消继承。

`position: absolute`——绝对定位，使用该属性能够很准确地将元素移动到想要的位置，它是相对独立出来的，丝毫不影响其他方向层。

`position: fixed`——相对于窗口固定定位，该元素的定位方式同 `absolute` 类似，但它的包含块是视区本身，在屏幕媒体（如 Web 浏览器）中，元素在文档滚动时不会再浏览器视察中移动。

`position: relative`——相对定位，是相对于元素默认的位置的定位。

10.12 提前进入，囤积自信（DIV 布局）

第 455 道：实现一个左右布局的结构，左侧自适应，右侧宽 280px。

解析：

```
<div id="wrap">
  <div id="content" style="height:100px;"> 左侧 </div>
  <div id="slidebar" style="height:100px;
    width:280px;"> 右侧    </div>
</div>
<style>
  #wrap{position:relative;}
  #slidebar{position:absolute; right:0; top:0;}
  #content{margin-right:280px;}
</style>
```

第 456 道：为什么利用多个域名来存储网站资源会更有效？

解析：

- CDN 缓存更方便；
- 突破浏览器并发限制；
- 节约 cookie 带宽；
- 节约主域名的连接数，优化页面响应速度；
- 防止不必要的安全问题。

第 457 道：在空白处加上 CSS，让这个 DIV 在页面居中（上下左右都居中）。

```
<div style=" border:1px solid black;width:200px;height:
200px;" >hello</div>
```

解析：

```
position:absolute;left:50%;top:50%;margin-left:-100px;
margin-top:-100px;
```

第 458 道：定义一个 DIV，在 IE8 下高度为 100px，在 IE7 下高度为 120px，在 IE6 下高度为 140px。

解析：

```
{padding-top:10px; height:90px;!Important; height:100px;}
```

IE6、IE7 和 IE8 对 DIV 高度的解析是不一样的，在 IE6 下，DIV 的高度和 padding、margin 没关系，而 IE7、IE8 则会受 padding 或者 margin 的影响。

第 459 道：如何居中 DIV ？

解析：

对需要水平居中的 DIV 层添加以下属性：

```
margin-left: auto; margin-right: auto;
```

主要的样式定义如下：

```
body{text-align: center; }#center{margin-right:auto; margin-left: auto; }
```

说明：

首先在父级元素定义 text-align: center；这个的意思就是将父级元素内的内容居中；对于 IE 来说，这样设定就已经可以了，但在 mozilla 中不能居中，解决办法就是在子元素定义时再加上 ‘margin-right:auto ; margin-left:auto;’ 需要说明的是，如果你想用这个方法使整个页面居中，建议不要套在一个 DIV 里，你可以依次拆出多个 DIV，只是在拆出每个 DIV 时定义 margin-right:auto; margin-left:auto; 就可以了。

第 460 道：下面关于 <div> 和 标记的描述，错误的是（ ）。

- A. <div> 标记默认情况下是块模式，即标记前后有类似换行符一样的功能
- B. 标记默认情况下是行模式，即标记前后内容在同行显示
- C. <div> 标记和 标记行块模式，可以通过 CSS 的 display 属性予以调整
- D. 前后两个 <div> 标记块的内容，不能出现在同一行

答案：D

解析：

<div> 简单而言是一个区块容器标记，即 <div> 与 </div> 之间相当于一个容器，可以容纳段落、标题、表格、图片，乃至章节、摘要和备注等各种 HTML 元素。因此，我们可以把 <div> 与 </div> 中的内容视为一个独立的对象，用于 CSS 控制。声明时只需要对 <div> 进行相应的控制，其中的各种标记元素会因此改变。

 标记与 <div> 标记一样，作为容器标记而被广泛应用在 HTML 语言中。在 与 </div> 中间同样可以容纳各种 HTML 元素，从而形成独立的对象，在上例中，如果把 “<div>” 替换成 “”，样式表中把 “div” 替换成 “span”，执行后会发现效果完全一样，可以说 <div> 与 这两个标记起

到的作用都是独立出各个区块，在这个意义上两者没有太多的区别，大家可以自行尝试一下。

<div> 标记与 标记的区别：<div> 标记是一个块级元素，包围的元素自动换行。而 标记仅仅是一个行内元素，在它的前台不会换行。 没有结构上的意义，纯粹是应用模式，当其他行内元素都不合适时，就可以使用 span 元素。此外， 标记可以包含于 <div> 标记之中，成为它的子元素，而反过来则不成立，即 标记不能包含 <div> 标记。

第 461 道：当 <p> 元素设置为 bold 时，以下哪条 CSS 语句的说法是正确的（ ）

- A. <p style="text-size:bold;"> B. <p style="font-size:bold;">
C. p{text-size:bold;} D. p{font-weight:bold;}

答案：D

解析：

font-weight 的属性包括 “normal” 和 “bold” 两个。

10.13 拥有别人没有的（盒子布局）

第 462 道：介绍 CSS 盒子模型。

解析：

如果 CSS 对 HTML 文档元素生成了一个描述该元素在 HTML 文档布局中所占空间的矩形元素框（element box），那么我们可以形象地将其看作是一个盒子。CSS 围绕这个盒子产生了一种“盒子模型”概念，通过定义一系列与盒子相关的属性（内容、填充、边框、边界），可以控制各个盒子乃至整个 HTML 文档的表现效果和布局结构。虽然 CSS 中没有盒子这个单独的属性对象，但它却是 CSS 中无处不在的一个重要组成部分。

第 463 道：列出 display 的值，并说明它们的作用？

解析：

block：像块元素一样显示。

none：隐藏元素并释放其位置。

inline-block：像行内元素一样显示，但其内容像块元素一样显示。

list-item：像块类型元素一样显示，并添加样式列表标记。

第 464 道：说说标准盒子模型和旧 IE 盒子模型有什么区别，以及触发不同模型的条件。

解析：

标准盒子模型是由 margin、border、padding、content 组成的，但 content 包

含其他部分（即 content 是一个独立的部分）；而旧 IE 盒子模型的组成部分为 margin、border、padding、content，它与标准盒子模型不同的是，旧 IE 盒子模型的 content 部分包含了 border 和 padding。

10.14 既然无处可逃，不如帅气迎接（CSS 排版）

第 465 道：网页设计采用 DIV+CSS 有什么好处？

解析：

- (1) 大大缩减页面代码。
- (2) 对搜索引擎更加友好，同时获得更好的网站排名。
- (3) 兼容性卓越。
- (4) 缩短网站改版时间。
- (5) 强大的控制和排版布局能力。
- (6) 提高易用性。
- (7) 扩展性能优越。
- (8) 灵活的页面布局。
- (9) 表现和内容相分离。

第 466 道：写出下面描述的 CSS 代码：

定义一个 <div> 标签，宽度为 300px，字体颜色是 #ff0，行高是 23px，边框是红色，字体大小是 12px，加粗，上下内边距为 5px，左右内边距为 10px，文字强制一行显示，超出的部分隐藏。

解析：

```
div{width:300px; color:#ff0; line-height:23px; border:1px
solid red; font-size:12px; font-weight:bold; padding:5px 10px
5px 10px;
white-space:nowrap; overflow:hidden;}
```

第 467 道：怎样使用一个层，使其垂直居中于浏览器中？

解析：

```
div {
    position:absolute;
    top:50%;
    left:50%;
    margin:-100px 0 0 -100px;
    width:200px;
    height:200px;
    border:1px solid red;
}
```

10.15 “限量版”的题（CSS 注释）

第 468 道：CSS 选择符有哪些？哪些属性可以继承？

解析：

- (1) 通配选择符，语法为：* { sRules }
- (2) 类型选择符，语法为：E { sRules } td { font-size:14px; width:120px; }
- (3) 属性选择符，语法为：

E [attr] { sRules }

E [attr = value] { sRules }

E [attr ~= value] { sRules }

E [attr |= value] { sRules }

示例：

h[title] { color: blue; }/* 所有具有 title 属性的 h 对象 */

span[class=demo] { color: red; }

div[speed="fast"][doron="no"] { color: red; }

a[rel~="copyright"] { color:black; }

(4) 包含选择符，语法为：E1 E2 { sRules } table td { font-size:14px; }

(5) 子对象选择符，语法为：E1 > E2 { sRules } div ul>li p { font-size:14px; }

(6) ID 选择符，语法为：#ID { sRules }

(7) 类选择符，语法为：E.className { sRules }

(8) 选择符分组，语法为：E1 , E2 , E3 { sRules }

(9) 伪类及伪对象选择符，语法为：

E : Pseudo-Classes { sRules }

(Pseudo-Classes)[:link :hover :active :visited :focus :first-child :first :left :right :lang]

E : Pseudo-Elements { sRules }

(Pseudo-Elements)[:first-letter :first-line :before :after]

可以继承的有：font-size、font-family、color。

不可继承的有：border、padding、margin、background-color、width、height 等。

第 469 道：在 CSS 的三种选择符中，哪个优先级最高（ ）

A. 标签选择符 B. 类选择符 C. ID 选择符

答案：C

解析：

- (1) 继承不如指定。
- (2) #ID>.class> 标签选择符。
- (3) 包含越具体越强大。

(4) 标签 #ID>#ID, 标签 .class>.class。

四大原则的权重是：原则一>原则二>原则三>原则四。

第 470 道：下列哪个不是 CSS 的选择符（ ）

A. p B. .id C. #box D. Shop

答案：D

解析：

p——类型选择符：类型选择符就是网页元素本身，定义时直接使用元素名称。

.id——class 选择符：在一个文档中可以为不同类型的元素定义相同的类名，class 可以实现把相同样式的元素统一为一类，使用 class 选择符时要先为每个元素定义一个 class 属性。

#box——ID 选择符：它是唯一的，不同元素的 ID 值是不能重复的，ID 选择符为每个元素的具体对象定义不同的样式，方便用户更加精细的控制页面。使用 ID 选择符时要先为每个元素定义一个 ID 属性。

10.16 值得分享的考题（iframe）

第 471 道：新的 HTML5 文档类型和字符集是？

解析：

HTML5 文档类型：

```
<!doctype html>
```

HTML5 使用 UTF-8 编码：

```
<meta charset=" UTF-8" >
```

第 11 章



出类拔萃，终成大器 [DIV+CSS高级面试题]

还在迷茫中吗？页面布局摆平一切！CSS 连接如同千军万马，更是使人心惊胆战；悬浮效果瞬间让你一步登天，位居榜首；CSS 内外边距如同华丽的装扮，吸人眼球。

11.1 程序“猿”你懂了吗（DOCTYPE）

第 472 道：<!DOCTYPE> 标签的定义与用法。

解析：

定义：<!DOCTYPE> 声明位于文档中最前面的位置，处于 <html> 标签之前。此标签可告知浏览器文档使用哪种 HTML 或 XHTML 规范。

该标签可声明三种 DTD 类型，分别表示严格版本、过渡版本以及基于框架的 HTML 版本。（假如文档中的标记不遵循 DOCTYPE 声明所指定的 DTD，这个文档除了不能通过代码校验之外，还有可能无法在浏览器中正确显示。）

用法：

（1）如果需要干净的标记，免于表现层的混乱，用 XHTML Strict DTD 类型。

（2）Transitional DTD 可包含 W3C 所期望移入样式表的呈现属性和元素。如果用户使用了不支持层叠样式表（CSS）的浏览器，以至于不得不使用 HTML 的呈现特性时，用 Transitional DTD 类型。

（3）Frameset DTD 被用于带有框架的文档。除 Frameset 元素取代了 body 元素之外，Frameset DTD 等同于 Transitional DTD。

第 473 道：什么是 DOCTYPE？你常用的 DOCTYPE 是什么？

解析：

DOCTYPE 是 DOCument TYPE（文档类型）的简写，用来说明所用的 XHTML 或者 HTML 是什么版本。

其中的 DTD（例如上例中的 xhtml1-transitional.dtd）为文档类型定义，里面包含了文档的规则，浏览器就根据定义的 DTD 来解释页面的标识，并展现出 <!DOCTYPE html>。

11.2 面试其实很简单，就看你了（CSS Hack）

第 474 道：介绍你所知道的 CSS Hack 技巧。

解析：

_width 针对于 IE6。*width、+width 针对于 IE6 和 IE7。

color: red\9; /*IE8 及以下版本浏览器 */（但是测试可以兼容到 IE10）

*+html 与 *html 是 IE 特有的标签，Firefox 暂不支持，而 *+html 又为 IE7 的特有标签（但是测试 *html 兼容 IE6，IE7，IE8，IE9，IE10。*+ 兼容 IE7，IE8，IE9，IE10）。

!important 在 IE 中会被忽视，IE6,IE7,IE8 不识别，IE9+（包括 IE9）是识别的。

第 475 道：下列 CSS Hack 的写法分别用于哪些浏览器。

- A. _ B. * 和 + C. \0 D. \9
E. \9\0 F. !important

解析：

A. _：用于 IE6。

B. * 和 +：用于 IE6，IE7。

C. \0：用于 IE8，IE9。

D. \9：用于 IE6，IE7，IE8，IE9。

E. \9\0：用于 IE9。

F. !important：除 IE6 不能识别外；Firefox，IE9，IE8，IE7 都能识别。

11.3 别让面试，输给了心情（悬浮效果）

第 476 道：请用 DIV+CSS 简单写出一段“返回顶部”的悬浮效果。

解析：

```
<div style="position:fixed; top:50%; left:50%; margin:-250px
0 0 -250px; height:500px; width:500px; z-index:999;">
  <a href="广告图链接"></a> </div>
```

11.4 面试不要瞎忙，不经意的才是最好的（CSS 优先级）

第 477 道：内联和 important 哪个优先级高？

解析：

内联的优先级最高。

第 478 道：请解释一下 CSS 的优先级，并说明优先级算法如何计算？

解析：

CSS 优先级包含四个级别：文内选择符、id 选择符、class 选择符、元素选择符。

CSS 优先级的计算规则如下：

元素标签中定义的样式（style 属性），加 1,0,0,0。

每个 id 选择符（如 #id），加 0,1,0,0。

每个 class 选择符（如 class）、每个属性选择符（如 [attribute=]）、每个伪类（如 hover）加 0,0,1,0。

每个元素选择符（如 p）或伪元素选择符（如：firstchild）等，加 0,0,0,1。

然后，将这四个数字分别累加，得到每个 CSS 定义的优先级的值。

接着，从左到右逐位比较大小，数字大的 CSS 样式的优先级就高。

注意：

（1）!important 声明的样式优先级最高，如果有冲突，那么再进行计算。

（2）如果优先级相同，则选择最后出现的样式。

（3）继承得到的样式的优先级最低。

第 479 道：如何解决 IE6 下 select 优先级高于 div 而导致的无法遮盖问题？

解析：

有两种解决办法：

（1）当浮动层 div 出现时，用 JS 将 select 隐藏，当浮动层 div 消失时，select 恢复出现。这种方法早期用得比较多，这里不多介绍。

（2）用 select 同级别元素——iframe 标签，将实际的浮动层 div 和 iframe 放在一起，间接地使 div 遮住 select。

第 480 道：CSS 中的 id 标签和 class 标签，哪个定义的级别高？

解析：

在结构相同情况下，id 优先级要高于 class。

id 是唯一的。class 在一个页面可以出现多个。

11.5 我不知将去何方，但我已在路上（定位）

第 481 道：position 都有哪些属性？fixed 是相对谁来定位的？

解析：

static：默认属性，静态定位。

relative：相对定位，相对于父元素的定位，需要配合 top、left、right、

bottom、z-index 等属性。

absolute：绝对定位，相对于父元素的定位，需要配合 top、left、right、bottom、z-index 等属性。

fixed：固定定位，相对于父元素的定位，需要配合 top、left、right、bottom、z-index 等属性（IE6 不支持）。

第 482 道：在 position 值中，relative 和 absolute 如何定位原始点？

解析：

相对定位，它是参照父级的原始点为原始点，无父级则以 body 的原始点为原始点，配合 TRBL 进行定位，当父级内有 padding 等 CSS 属性时，前级的原始点则参照父级内容区的原始点进行定位。

绝对定位，它是参照浏览器的左上角，配合 top、right、bottom、left（下面简称 TRBL）进行定位，在没有设定 TRBL 时，默认依据父级的坐标原始点为原始点。如果设定了 TRBL，并且父级没有设定 position 属性，那么当前的 absolute 则以浏览器左上角为原始点进行定位，位置将由 TRBL 决定。

第 483 道：定位的值有几种？区别是什么？

解析：

static：默认值，没有定位，元素出现在正常的流中（忽略 top、bottom、left、right 或者 z-index 声明）。

relative：生成相对定位的元素，相对于其正常位置进行定位。元素的位置通过 left、top、right，以及 bottom 属性进行规定。

fixed：元素框的表现类似于将 position 设置为 absolute，不过其包含块是视窗本身。

absolute：生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。元素的位置通过 left、top、right，以及 bottom 属性进行规定。

第 484 道：CSS 中的 position 属性有几种取值？分别说一下它们的作用。

解析：

position 属性取值：static、relative、absolute、fixed、inherit。

postision：static，始终处于文档流给予的位置。看起来好像没有用，但它可以快速取消定位，让 top、right、bottom、left 的值失效。在切换时可以尝试这个方法。

除了 static 值，在其他三个值的设置下，z-index 才会起作用。（确切地说，z-index 只在定位元素上有效）。

position：relative 和 position：absolute 都可以用于定位，区别在于前者的 DIV 还属于正常的文档流，后者已经脱离了正常文档流，不占据空间位置，不会将父类撑开。定位原点 relative 是相对于它在正常文档流中的默认位置偏移，它原本占据的空间任然保留；absolute 相对于第一个 position 属性值不为 static 的

父类。所以设置了 `position: absolute`，其父类的该属性值要注意，而且 `overflow: hidden` 也不能乱设置，因为不属于正常文档流，不会占据父类的高度，也就不会有滚动条。

`fixed` 的旧版本不支持 IE，却是很有用的。其定位原点相对于浏览器窗口，而且不能变。常用于 `header`、`footer`，或者一些固定的悬浮 `div`，随滚动条滚动既稳定又流畅，比 JS 好多了。`fixed` 可以有很多创造性的布局和作用，但兼容性是个问题。

`position: inherit`。规定从父类继承 `position` 属性的值，所以这个属性也是有继承性的，但是任何版本的 IE 都不支持该属性值。

11.6 面试如一道弧线，却能摆平一切（页面布局）

第 485 道：在定义通用样式时，经常会使用 `ul{margin:0;padding:0}` 的写法，这样做的目的是什么？

解析：

删除浏览器这些默认值，方便后面的设置。

第 486 道：判断题：CSS 属性 `font-style` 用于设置字体的粗细。

答案：错。

解析：

`font_weight` 用于设置字体的粗细。

第 487 道：在 HTML 页面布局中，`position` 的默认值是什么？

解析：

`static`

第 488 道：制作一个宽为 800px，高为 500px 的 DIV，使其在页面上下左右绝对居中。

解析：

```
<div style="width:800px; height:500px; position:absolute;
top:50%; left:50%; margin-left:-400px; margin-top:-250px;
background:red;">
```

第 489 道：一个宽度不确定的 DIV 里面放三个水平对齐的 DIV，左右两个 DIV 宽度固定为 150px，中间那个 DIV 充满剩余的宽度。

解析：

```
<div style="width:500px;position:absolute;left:200px;top:100px;" >
  <div id="px1" style="position:absolute; left:0px;height:100px;
width:150px; background-color:green;"></div>
  <div id="px2" style="float:left; height:100px; width:100%;
```



```
background-color:red;"></div>
    <div id="px3" style="position:absolute; right:0px;height:
100px; width:150px; background-color:gray;"></div>
</div>
```

11.7 程序员，我不知道你心里是怎么想的（CSS 字体）

第 490 道：判断题：如果浏览者没有安装网页上所设置的文字，则会以默认的字取代原来的。

答案：对。

解析：

@font-face 版本：CSS2 兼容性：IE4+

语法：@font-face { font-family : name ; src : url (url) ; sRules }

取值：name——字体名称。任何可能的 font-family 属性的值。

url (url)——使用绝对或相对 url 地址指定 OpenType 字体文件。

sRules——样式表定义。

说明：设置嵌入 HTML 文档的字体。此规则无默认值。

此规则让你能够在网页上使用客户端本地系统上可能没有的字体。url 地址必须指向 OpenType 字体文件（.eot 或 .otf）。此文件包含可以转换为 TrueType 字体的压缩字体数据，可以用来提供 HTML 文档使用该字体，或取代客户端系统已有的同名字体。此文件可以使用 Microsoft WEFT 工具制作。

11.8 面试时谁没有耐心，谁就没有智慧（CSS 表格）

第 491 道：在表单中包含性别选项，且默认状态为“男”被选中，下列正确的是（ ）

- A. <input type=radio name=sex checked> 男
- B. <input type=radio name=sex enabled> 男
- C. <input type=checkbox name=sex checked> 男
- D. <input type=checkbox name=sex enabled> 男

答案：A

解析：

单选框被选中为 checked。

第 492 道：collpadding td 标签中的 colspan、rowspan 分别起什么作用？

解析：

colspan 是跨列，rowspan 是跨行，可以看作是网页设计表格中的列和行的一个属性。

跨列相当于把两列或者多列合并成一个单元格。

同理，跨行是把两行或者多行合并成一行。

colspan 和 rowspan 分别作用于网页设计表格的 `<td></td>`、`<tr></tr>` 中。

两者的作用都是合并，其主要区别就是，一个是水平方向可以合并列，即 colspan；一个是垂直方向可以合并行，即 rowspan。

11.9 面试，你紧张了吗（CSS 内外边距）

第 493 道：分别用什么改变元素的外边距和改变元素的内填充？

解析：

margin 和 padding

第 494 道（判断题）：一个大的 DIV 块里包含一个小的 DIV，设置小的 DIV 与大的 DIV 的左边距为 5px，该样式的标准写法是 `margin-left:5px;`；

答案：错。

解析：

给大盒子设置成 `padding_left:5px;`

11.10 不能白看，看完必过（CSS 文本）

第 495 道：IE 中如何让超出宽度的文字显示为省略号？

解析：

```
div.titleholder {
    font-family: ms sans serif, arial;
    font-size: 8pt;
    width: 100;
    text-overflow: ellipsis;
    overflow: hidden;
    white-space: nowrap;
}
```

第 496 道（判断题）：CSS 中的属性 overflow 用于设置元素超过宽度时，是否隐藏或显示滚动条？

答案：对。

解析：

overflow 超出的部分会被隐藏，并且当我们给 DIV 加上 overflow: hidden 属性时，其中的带浮动属性的 DIV 在这个立体的浮动中已经被清除了。

第 497 道：请写一下首行缩进两个字符的 CSS 代码及常见的用法。

解析：

text-indent:2em;

text 的意思是文本；indent 在计算机英语中的意思是缩进；em 的意思是相对文字大小，1em 就是一个文字大小，2em 就是两个文字大小。

第 498 道：<style type="text/css">

```
#text{color : red}
p#text{color :brown}
p.text{color : green}
p{color : while}
</style>
```

<p id="text" class="text">请说出我的颜色</p>

请问 p 标签的文本颜色是什么？

答案：red

解析：

#text{color:red} 的优先级最高。

第 499 道：为什么在 Firefox 下文本无法撑开容器的高度？

解析：

标准浏览器中固定高度值的容器是不会像 IE6 那样被撑开的，那么当我们既想设置为固定高度，又想能被撑开时，需要怎样设置呢？办法就是去掉 height，设置 min-height:200px; 这里为了照顾不认识 min-height 的 IE6，可以这样定义：

```
{
    height:auto!important;
    height:200px;
    min-height:200px;
}
```

第 500 道：请写一段 CSS 代码，让 Chrome 支持小于 12px 的文字。

解析：

html,body{-webkit-text-size-adjust:none;} 或 .divcss5{-webkit-text-size-adjust:none;}

11.11 面试如同千军万马，更是使人心惊胆战（CSS 链接）

第 501 道：DIV CSS 设计中如何去掉链接的虚线框？

解析：

```
a:focus {
    outline:none;
    -moz-outline:none;
}
```

第 502 道：以下链接到电子邮件的正确格式是（ ）

- A. 邮箱
- B. 邮箱
- C. 邮箱
- D. 邮箱

答案：A

解析：

我们要在网页中创建一个形如“mailto: aaa@5icool.org”这样的超级链接，用鼠标单击一下该超级链接，浏览器会自动调用系统默认的邮件客户端程序，同时在邮件编辑窗口的收件人设置栏中自动写上收件人的地址，而其他的内容都是空白的，留给访问者自行填写。

第 503 道：在 CSS 中，下面哪种方法表示超链接文字在鼠标经过时，超链接文字无下划线？（ ）

- A. A:link{text-decoratton:underline}
- B. A:hover{text-decoratton:none}
- C. A:active{text-decoratton:blink}
- D. A:visited{text-decoratton:overline}

答案：B

解析：

text-decoratton 是文字修饰效果的意思；none 参数表示超链接文字不显示下画线。

第 504 道：下面哪一个属性可以在新窗口打开一个链接（ ）

- A. _parent
- B. _blank
- C. _self
- D. _top

答案：B

解析：

_blank 表示在新的窗口打开链接。

target 属性值一般默认是 _self，表示在原来的窗口打开链接，只要把 target 的属性值改为 _blank 就可以了。

第 505 道（判断）：超链接只能在不同的网页之间进行跳转。

答案：错

解析：

添加超链接 点此跳转 ，就可以跳转到

a.html 了。

第 506 道：怎样解决超链接访问过后 hover 样式不出现的问题？

解析：

```
<style type="text/css">
  <!--
    a:link {}
    a:visited {}
    a:hover {}
    a:active {}
  -->
</style>
```

第 507 道：在新窗口打开链接的方法是_____。

答案：

target=_bank

第 12 章



炉火纯青，大杀四方 [DIV+CSS终极面试题]

说得简单做起来难，三层构成秒杀全场。群压四方的兼容问题，更是需要万众一心，CSS Sprites 可遇而不可求，CSS 浮动多美好。

12.1 面试就像自行车，说得简单，其实还要靠自己（三层构成）

第 508 道：前端页面有哪三层构成，分别是什么？作用是什么？

解析：

(1) 网页的结构层 (structural layer)，由 HTML 或 XHTML 之类的标记语言负责创建。标签，也就是那些出现在尖括号里的单词，对网页内容的语义含义做出了描述，但这些标签不包含任何关于如何显示有关内容的信息。例如，P 标签表达“这是一个文本段。”了这样一种语义。

(2) 网页的表示层 (presentation layer)，由 CSS 负责创建。CSS 对“如何显示有关内容”的问题做出了回答。

(3) 网页的行为层 (behavior layer)，负责回答“内容应该如何对事件做出反应”这一问题。这是 JavaScript 语言和 DOM 主宰的领域。

12.2 面试就算终有一散，也别辜负相遇（CSS Sprites）

第 509 道：谈谈对 CSS Sprites 技术优缺点的理解。

解析：

1. CSS Sprites 的优点

(1) 利用 CSS Sprites 能很好地减少网页的 http 请求，从而大大提高了页面的性能，这也是 CSS Sprites 最大的优点，也是其被广泛传播和应用的主要原因；

(2) CSS Sprites 能减少图片的字节，曾经多次比较过，把 3 张图片合并成 1 张图片的字节总是小于这 3 张图片的字节总和。

2. CSS Sprites 缺点：

(1) 在图片合并时，要把多张图片有序的、合理的合并成一张图片，还

要留好足够的空间，防止板块内出现不必要的背景。这些还好，最痛苦的是，在宽屏及高分辨率的屏幕下的自适应页面，如果图片不够宽，很容易出现背景断裂；

(2) CSS Sprites 在开发时比较麻烦，要通过 Photoshop 或其他工具测量计算每一个背景单元的精确位置，这是“针线活”，没什么难度，但是很烦琐。幸好腾讯的鬼哥用 RIA 开发了一个 CSS Sprites 样式生成工具，虽然在一些使用上还不够灵活，但是已经比 Photoshop 测量方便多了，而且样式直接生成，只需复制就 OK！

(3) CSS Sprites 在维护时比较麻烦，如果页面背景有少许改动，一般就要改这张合并的图片，所以无需改的地方最好不要动，这样避免改动更多的 CSS。如果在原来的地方放不下，那么只能（最好）往下加图片，此时图片的字节就增加了，且还要改动 CSS。

第 510 道：DIV+CSS 的布局与 table 布局相比有什么优点？

解析：

- (1) 改版的时候更方便，只要改 CSS 文件。
- (2) 页面加载速度更快，结构化清晰，页面显示简洁。
- (3) 表现与结构相分离。
- (4) 易于优化，搜索引擎（SEO）更友好，排名更容易靠前。

第 511 道：如何使用 CSS Sprites？

解析：

HTML 结构：

```
<ul>
    <li id="nav_1"></li>
    <li id="nav_2"></li>
    <li id="nav_3"></li>
</ul>
```

CSS 代码：

```
<style>
    #nav_1,#nav_2,#nav_3{width:80px;height:24px;
    background:url(images/nav.gif) no-repeat;}
    #nav_1{background-position: -5px -10px;}
    #nav_2{background-position: -105px -10px;}
    #nav_3{background-position: -210px -10px;}
</style>
```

12.3 面试短短的话语，却包含万千（CSS 中 a 的伪类）

第 512 道：CSS 的伪类有哪些？有什么作用？在各个浏览器下都兼容吗？

解析：

在 W3C 规范中，伪类有：active、hover、link、visited。

[CSS1] 所有主流浏览器都支持以上伪类，但是只对 <a href> 标签上的伪类支持最好。

[CSS2] 有 3 个 focus：如果规定了 <!DOCTYPE>，将在主流浏览器 &IE8+ 中支持（否则 IE 不支持）。

first-child（首个子对象）：必须声明 <!DOCTYPE> 才能够确保在 IE 中正常支持。

lang（语言）：如果规定了 <!DOCTYPE>，将在主流浏览器 &IE8+ 中支持（否则 IE 不支持）。

第 513 道：列举 CSS 中 a 的伪类，并描述他们的定义顺序？

解析：

在 CSS 中定义 a:link、a:visited、a:hover、a:active 顺序。

a:link、a:hover、a:active、a:visited 这几个元素，定义 CSS 时顺序不同，也会直接导致链接显示的效果不同。

举例来说：想让未访问链接颜色为蓝色，活动链接为绿色，已访问链接为红色。

第一种情况：定义的顺序是 a:visited、a:hover、a:link，这时会发现：把鼠标放到未访问过的蓝色链接上时，它并不变成绿色，只有放在已访问的红色链接上，链接才会变绿。

第二种情况：把 CSS 定义顺序调整为 a:link、a:visited、a:hover，这时，无论鼠标经过的链接有没有被访问过，它都会变成绿色。这是因为，一个鼠标经过的未访问链接同时拥有 a:link、a:hover 两种属性，在第一种情况下，a:link 离它最近，所以它优先满足 a:link，而放弃 a:hover 的重复定义。在第二种情况下，无论链接有没有被访问过，它首先要检查是否符合 a:hover 的标准（即是否有鼠标经过它）。如果满足，则变成绿色；如果不满足，则继续向上查找，一直找到满足条件的定义为止。

第 514 道：CSS 新增伪类有哪些？

解析：

常用于控制表单控件有 :enabled、:disabled 和 :checked。

其中，:enabled、:disabled 表示控制表单控件的禁用状态。

:checked 表示单选框或复选框被选中。

12.4 面试要勇敢，前方的路很长（CSS 浏览器兼容）

第 515 道：你知道哪些 CSS 浏览器兼容性问题。

解析：

(1) div 的垂直居中问题。vertical-align:middle; 将行距增加到和整个 div 一样高 line-height:200px; 然后插入文字，就垂直居中了。缺点是要控制内容不要换行。

(2) margin 加倍的问题。设置为 float 的 div 在 IE 下设置的 margin 会加倍。这是一个 IE6 都存在的 Bug。解决方案是在这个 div 里面加上 display:inline;。例如：`<#div id="imfloat">` 相应的 CSS 为 `#IamFloat{ float:left; margin:5px; /*IE 下理解为 10px*/display:inline; /*IE 下再理解为 5px*/ }`。

(3) 浮动 IE 产生的双倍距离 `#box{ float:left; width:100px; margin:0 0 100px; // 这种情况之下 IE 会产生 200px 的距离 display:inline;` 这里细说一下 block 与 inline 两个元素。block 元素（块元素）的特点是，总是在新行上开始，高度，宽度，行高，边距都可以控制；Inline 元素（内嵌元素）的特点是，和其他元素在同一行上，不可控制；`#box{ display:block; // 可以为内嵌元素模拟为块元素 display:inline; // 实现同一行排列的效果 display:table;`

(4) IE 与宽度和高度的问题。IE 不认得 min 这个定义，但实际上它把正常的 width 和 height 当作有 min 的情况来使。这样问题就大了，如果只用宽度和高度，正常的浏览器里这两个值就不会变，如果只用 min-width 和 min-height 的话，在 IE 下等于没有设置宽度和高度。比如要设置背景图片，这个宽度是比较重要的。要解决这个问题，可以这样：`#box{ width: 80px; height: 35px;}html>body #box{ width: auto; height: auto; min-width: 80px; min-height: 35px;}`

第 516 道：请列举 IE6 的一些 Bug 的解决方法。

解析：

1. IE6 双倍边距 Bug

当页面上的元素使用 float 浮动时，不管是向左浮动还是向右浮动，只要该元素带有 margin 像素，就会使该值乘以 2，例如，在 IE6 中，“margin-left:10px” 的值就会被解析为 20px。想要解决这个 Bug 就需要在该元素中加入 display:inline 或 display:block，明确其元素类型即可解决双倍边距的 Bug。

2. IE6 中 3px 问题

当元素使用 float 浮动后，元素与相邻的元素之间会产生 3px 的间隙。诡异的是，如果在相邻容器的内部，右侧的容器没设置高度时出现 3px 的间隙，那么当设定高度后又跑到容器的相反侧了。要解决这类 Bug 的话，需要使布局在同一行的元素都加上 float 浮动。

3. IE6 中奇数高宽的 Bug

IE6 中奇数的高宽显示大小与偶数高宽显示大小存在一定的不同。其主要问题出在奇数高宽上。要解决此类问题，只需要尽量将外部定位的 div 高宽写成偶数即可。

4. IE6 中图片链接的下方有间隙

IE6 中图片的下方会存在一定的间隙，尤其在图片垂直挨着图片的时候，即可看到这样的间隙。要解决此类问题，需要将 `img` 标签定义为 `display:block` 或定义 `vertical-align` 对应的属性，也可以为 `img` 对应的样式写入 `font-size:0`。

5. IE6 中空元素的高度 Bug

如果一个元素中没有任何内容，当在样式中为这个元素设置了 `0 ~ 19px` 之间的高度时，此元素的高度始终为 `19px`。

解决的方法有 4 种：

- 在元素的 CSS 中加入：`overflow:hidden`；
- 在元素中插入 HTML 注释：`<!-->`；
- 在元素中插入 HTML 的空白符：` `；
- 在元素的 CSS 中加入：`font-size:0`。

6. 重复文字的 Bug

在某些比较复杂的排版中，有时候浮动元素的最后一些字符会出现在 `clear` 清除元素的下面。

解决方法如下：

- 确保元素都带有 `display:inline`
- 在最后一个元素上使用 `margin-right:-3px`
- 为浮动元素的最后一个条目加上条件注释：`<!-- [if !IE]>xxx<![endif]-->`
- 在容器的最后元素使用一个空白的 `div`，为这个 `div` 指定不超过容器的宽度

7. IE6 中 `z-index` 失效

具体 Bug 为，如果元素的父级元素设置的 `z-index` 为 1，那么其子级元素再设置 `z-index` 时会失效，其层级会继承父级元素的设置，造成某些层级调整上的 Bug。

第 517 道：请尽可能多列举 IE6、IE7、Firefox 浏览器的常见 CSS 兼容性差别，以及解决方法？

解析：

1. 在 Firefox 下给 `div` 设置 `padding` 后会导致 `width` 和 `height` 增加，但 IE 不会。可用 `!important` 解决，如 `width:115px! important; width: 120px; padding: 5px;`；必须注意的是，`!important` 一定要在前面。

2. 居中问题

(1) 垂直居中。将 `line-height` 设置为与当前 `div` 相同的高度，再使用 `vertical-align: middle`（注意内容不要换行）。

(2) 水平居中。可使用 `margin: 0 auto`；（当然不是万能）。

(3) 若需给 `a` 标签的内容加上样式，需要设置 `display: block`；（常见于导航

标签)。

(4) Firefox 和 IE 对 box 理解的差异，导致了相差 2px，还有设为 float 的 div 在 IE 下 margin 加倍等问题。

(5) ul 标签在 Firefox 下面默认有 list-style 和 padding。最好事先声明，以免引起不必要的麻烦（常见于导航标签和内容列表）。

(6) 作为外部 wrapper 的 div 不要定死高度，最好还加上 overflow: hidden 以达到高度自适应。

(7) 关于手形光标 cursor: pointer，而 hand 只适用于 IE。

使用兼容代码是兼容最推荐的模式。

第 518 道：一个网页制作完成后，在发布之前，我们会对页面进行测试，测试内容主要包括哪几个方面？

解析：

测试内容主要包括以下几个方面。

- (1) 页面效果是否美观。
- (2) 链接是否完好。
- (3) 页面功能（如验证、交互等）是否正确。
- (4) 测试不同浏览器的兼容性。

第 519 道：在网页设计与制作时，为了使制作出来的网页下载速度快、布局合理、浏览方便、和谐悦目，应注意哪些问题？

解析：

- (1) 网页文件大小。
- (2) 页面布局。
- (3) 页面导航。
- (4) 图像大小。
- (5) 颜色搭配。
- (6) 背景图像。

第 520 道：IE6 下为什么无法定义 1px 左右高度的容器？

解析：

IE6 下出现这个问题是由默认的行高造成的，解决的方法有很多，例如：
overflow: hidden | zoom: 0.08 | line-height: 1px。

第 521 道：Firefox 嵌套 DIV 标签的居中问题有什么解决方法？假定有如下情况：

```
<div>
  <div id=" b" ></div>
</div>
```

要实现 b 在 a 中居中放置，一般只需要在 CSS 中设置 a 的 text-align 属性为

center，此方法在 IE 里看起来一切正常，但是在 Firefox 中 b 却是居左的。

解析：

解决办法就是设置 b 的横向 margin 为 auto。例如，设置 b 的 CSS 样式为：
margin:0 auto;。

第 522 道：如何用 CSS 分别定义 IE6、IE7、IE8 的 width 属性？

解析：

```
_width:6px;           /*IE6*/
+width:7px;           /*IE7*/
width:8px\0;          /*IE8*/
```

12.5 程序“猿”看完就乐了（CSS 水平对齐）

第 523 道：一个盒子里面一张图片水平且垂直居中（盒子大小随意），试写出它的样式和 HTML。

解析：

```
<!DOCTYPE html>
<style>
    div {
        width:200px;height:200px;
        border:1px solid #CCC;
        overflow:hidden;
        /* 居中实现 */
        text-align:center;
        font-size:0px;
        line-height:200px;
        /* 兼容 IE6 */
        _font-size:178px;
    }
    img {vertical-align:middle;}
</style>
<div>
    .
</div>
```

12.6 “面试”是一件多么美的事（CSS 浮动）

第 524 道：简单解释一下浮动和它的工作原理，并提供一下清除浮动的技巧？

解析：

浮动是指浮动元素脱离文档流，不占据空间。浮动元素碰到包含它的边框

或者浮动元素的边框时停留。

技巧：

(1) 使用空标签清除浮动。这种方法是在所有浮动标签后面添加一个空标签，定义 `css clear:both`。弊端就是增加了无意义标签。

(2) 使用 `overflow`。给包含浮动元素的父标签添加 CSS 属性 `overflow:auto`; `zoom:1`，其中 `zoom:1` 用于兼容 IE6。

(3) 使用 `after` 伪对象清除浮动。该方法只适用于非 IE 浏览器。使用中需注意以下几点：

- 该方法中必须为需要清除浮动元素的伪对象设置 `height:0`，否则该元素会比实际高出若干像素；
- `content` 属性是必须的，但其值可以为空，讨论该方法时把 `content` 属性的值设为 “.”，但为空也是可以的。

第 525 道：请编写一个 `div` 的 CSS，使这个 `div` 能够居中浮动显示。

解析：

```
div{
    position:absolute;
    height:200px;
    width:200px;
    left:50%;
    top:50%;
    margin-left:-100px;
    margin-top:-100px;
}
```

第 526 道：当内部容器使用了 `float:left` 后，你会用哪些方法对父容器清除浮动，并谈一下各自优缺点？

解析：

1. 使用 `overflow` 清除

注意兼容问题：Block Formatting Contexts 概念是在 CSS 2.1 规范内被提出的。因此在 IE6/IE7 中并不被支持，这是由于之前的 IE 版本仅仅实现了 CSS 1 规范标准，以及一部分 CSS 2.0 规范标准。在 IE7 中，`overflow` 值为非 `visible` 时，可以触发 `hasLayout` 特性，这使得 IE7 同样可以使容器包含浮动元素。

2. 使用 `display:table` 清除

注意兼容问题：除去 Block Formatting Contexts 在 IE6/IE7 中的兼容性外，`display:table` 系列样式设定也不在 IE6/IE7 的支持范围之内。

3. 使用 `float:left` 和 `float:right`;

虽然这种方式并没有兼容问题，但在实际使用中并不推荐。因为很容易推断出，页面中只要有一个浮动元素，使用该方法清理浮动就不可避免地使页面

所有元素都浮动才可以达到预期效果。

12.7 面试是不可缺少的美好亮点（CSS 优势）

第 527 道：总结学习 CSS 容易出现错误的内容。

解析：

(1) 对于布局标签的定义。有时候写好了 CSS，但是反复调试却发现效果差强人意，总有那么一块空白，挥之不去。其实这就是对于标签定义的不严谨造成的。因为在 XHTML 的部分标签里，有一些默认属性值不一定为空或零。

(2) 缩写和大小写问题。CSS 对于 class 和 id 是区分大小写的，所以当样式不生效时，建议先检查一下大小写问题。

(3) 少用限定，多用继承。

(4) 多重 class 及就近优先原则。

(5) 链接的正确写法 :link :visited :hover :active。

第 528 道：什么是 CSS？它能做些什么？

解析：

CSS 就是一种叫作样式表 (Stylesheet) 的技术。也有的人称之为层叠样式表 (Cascading Stylesheet)。在主页制作时采用 CSS 技术，可以有效地对页面的布局、字体、颜色、背景和其他效果实现更加精确的控制。只要对相应的代码做一些简单的修改，就可以改变同一页面的不同部分，或者不同页数的网页的外观和格式。

它的作用：

(1) 在几乎所有的浏览器上都可以使用。

(2) 以前必须通过图片转换实现的功能，现在只要用 CSS 就可以轻松实现，从而更快地下载页面。

(3) 使页面的字体变得更漂亮，更容易编排，使页面真正赏心悦目。

(4) 可以轻松地控制页面的布局。

(5) 可以将许多网页的风格格式同时更新，不用再一页一页地更新了。

12.8 你可知道面试的重要（IE6 常见问题）

第 529 道：现在我们使用的 IE 的版本一般是多少，IE 的英文全称是什么？

解析：

目前 IE 的最高版本是 IE11，英文全称是：Internet Explorer。现在 IE Internet Explorer 6.0 SP2/5.5/5.0/6.0 SP1/7.0 beta2/7.0 beta3 都算对。IE12 正在研发中，正

式发布还需要很久。

第 530 道：简述 IE6、IE8、IE11 的区别。

解析：

1. 盒子模型

- IE6：（使用 !DOCTYPE 声明指定 standards-compliant 模式）， $\text{margin-left} + \text{border-left} + \text{padding-left} + \text{width} + \text{padding-right} + \text{border-right} + \text{margin-right}$ ，如果没有声明为标准模式，则为： $\text{margin-left} + \text{width} + \text{margin-right}$ 。
- IE8：同 IE6 的标准模式。
- IE11：同 IE6 的标准模式，但 IE11 遵循 CSS3 标准。

2. 内容宽度超过容器的宽度

- IE6：内容会撑开容器的宽度。
 - IE8：内容会溢出，容器不会被撑开。
 - IE11：同 IE8。
3. 容器内的内容有浮动属性，容器没有设置浮动且宽度超过内容的宽度
- IE6：容器的高度会被内容撑开。
 - IE8：容器高度不会被撑开。
 - IE11：同 IE8。

第 531 道：如何解决 IE6 的双倍边距 Bug？

解析：

给当前元素添加样式，使当前元素不为块，如：`display:inline;display:list-item`；这样当元素浮动时就不会在 IE6 下面产生双倍边距的问题了。

第 532 道：在 IE6 中无法定义图片的透明，怎么解决？

解析：

只需要在 png 背景的元素 CSS 中添加微软特有的滤镜就可以了。例如：

`_background:none;filter:progid:DXImageTransform.Microsoft.AlphaImageLoader (src=' 图片路径 ');`

需要注意：此图片路径若为相对路径，则是 HTML 文件的相对路径，而不是 CSS 文件的相对路径。

第 533 道：请写出 5 条 Firefox 和 IE 的脚本兼容问题。

解析：

- (1) 正式表达式问题。
- (2) 移除 Select 的条目。
- (3) `showModalDialog`。
- (4) `childNodes`。
- (5) `removeChild`。

(6) script 跨域问题。

综合提升

第 534 道：如何实现上、左、右 div 满屏布局？

解析：

```
<html>
<style>
*{ margin:0; padding:0;}
body{ width:100%;}
#div1{width:100%; height:150px;margin:0 auto; background:
#F00;}
#box{width:100%; margin:0 auto; height:300px;}
#box #div2{width:20%; float:left; background-color:#00F;
height:300px;}
#box #div3{ width:78%; float:right; height:300px; background-
color:#F60;}
</style>
<body>
<div id="div1"></div>
<div id="box">
<div id="div2"></div>
<div id="div3"></div>
</div>
</body>
</html>
```

第 535 道：在网页设计中 DIV 标记有何作用？

解析：

DIV 标记是一个区块级的 HTML 标记，在该标记之间可以添加段落、表格、图片等内容，使同一个 DIV 标记中的元素具有相同的样式，并可在页面显示时同时出现、移动及隐藏。

它的主要作用如下。

- (1) 将一些标记元素组织起来，应用 DIV 的属性为这些标记元素定义统一的样式；
- (2) 利用其 z-index 属性，实现页面内各元素的重叠显示效果；
- (3) 使页面上显示更多的特效功能。

第 536 道：如果一个网页需要同时有多种语言文字，那么网页需要什么编码格式。

解析：

UTF-8 是世界通用的语言编码格式。

网页使用 UTF-8 编码唯一的好处是，无论操作系统使用的是简体中文（GB2312 字符集）、繁体中文（BIG5 字符集），或者是朝鲜文、日文、法文、德文、俄文、阿拉伯文、希伯来文、西班牙文、葡萄牙文等各种语言文字，在使用这些语言文字时，都可以正常显示在网页中，其他任何人浏览时都会正常显示，不会有乱码，也不会有重码或字符冲突，不需要调整页面的语言编码设置即可正常浏览，多种语言字符可以同时共存在页面上。

第 537 道：左侧固定宽度为 200px，右侧宽度自适应浏览器分辨率，试写出 DIV 的样式和 HTML。

解析：

```
<div>
    <div id="right";style="height:100px;width:200px;
    float:right;background-color:red; "></div>
    <div id="left"style="height:100px;margin-right:200px;
    background-color:green;"></div>
</div>
```

第 538 道：CSS+DIV 开发 Web 页面的优势有哪些？

解析：

(1) CSS+DIV，在这个网页设计模式中，DIV 承担了网页的内容，CSS 承担了网页的样式。这样就使网页的内容和样式分离开来，有利于页面的维护升级。

(2) 有助于提高搜索引擎亲和力（快速找到需要的数据，而不像在 table 中需要一层层地查找）。

(3) 有助于页面的重构（如在 blog 中更换皮肤，直接套用另外一套样式就可以实现，而不用改动网页脚本）。

第 539 道：如何用 CSS 分别单独定义 IE6、IE7、IE8 的 width 属性（举例说明）。

解析：

```
height: 100px; /* 所有浏览器通用 */
height: 100px !important; /* IE7 和 Firefox 共用 */
color: #090\9; /* IE6、IE7、IE8 */
+color: #f00; /* 只针对 IE7 */
color: #f00; /* IE6、IE7 */
_color: #ff0; /* IE6 */
```

第 540 道：目前常用的 Web 标准静态页面语言是_____。

答案：HTML

解析：

常用的标记语言有：SGML（标准通用标记语言，元语言）、HTML（超文本标记语言）、XML（扩展标记语言，定义数据结构，适合数据传输）。

第 541 道：合理的页面布局中常听过结构与表现分离，那么结构是 _____，表现是 _____。

答案：DIV，CSS 样式。

解析：

在网页中，网页的结构就是 DIV，表现形式就是 CSS 样式。

第 542 道：最常遇到的兼容 Bug 有哪些？有哪些问题解决起来是最麻烦的？

解析：

(1) 不同浏览器的标签默认的外补丁和内补丁不同。

(2) 在块属性标签 float 后，又有横行的 margin 情况下，IE6 显示的 margin 比设置的大。

(3) 设置较小高度标签（一般小于 10px），在 IE6、IE7、遨游中，高度超出自己设置的高度。

(4) 行内属性标签，设置 display:block 后采用 float 布局，又有横行的 margin 的情况下，IE6 间距会产生 Bug（类似第二种）。

(5) 图片默认有间距。

(6) 标签最低高度设置 min-height 不兼容。

(7) 透明度的兼容 CSS 设置。

其中，不同浏览器的标签默认补丁，以及 IE6 png、IE6 fixed 的兼容问题不是很好解决。

第 543 道：在 Web 标准中，为什么 IE 无法设置滚动条颜色？

解析：

将 body 换成 html，即：

```
html {
  scrollbar-face-color:#f6f6f6;
  scrollbar-highlight-color:#fff;
  scrollbar-shadow-color:#eeeeee;
  scrollbar-3dlight-color:#eeeeee;
  scrollbar-arrow-color:#000;
  scrollbar-track-color:#fff;
  scrollbar-darkshadow-color:#fff;
}
```

第 544 道：DOCTYPE 的作用？严格模式与混杂模式的区别，如何触发这两种模式，区分它们有何意义？

解析：

DOCTYPE 是 DOcument TYPE（文档类型）的简写，是一组机器可读的规则，它们指示 (X)HTML 文档中允许有什么，不允许有什么，DOCTYPE 正是用

来告诉浏览器使用哪种 DTD，一般放在 (X)HTML 文档开头，用以告诉其他人这个文档的类型风格。

区别：严格模式是浏览器根据 Web 标准去解析页面，是一种要求严格的 DTD，不允许使用任何表现层的语法，如 `
`，混杂模式则是一种向后兼容的解析方法。

触发：根据不同的 DTD 触发，如果没有声明，那么默认为混杂模式。

第4篇

江湖路 ——jQuery 任逍遥

， 无尽头

jQuery 是 JavaScript 轻量级的框架（库），它简化了 HTML 文档遍历、事件处理、动画，并能轻松处理 Ajax 交互。在本篇，您将学习到如何选取 HTML 元素，以及如何对它们执行类似隐藏、移动，以及操作其内容等知识。

第 13 章



刻苦学艺，心无旁骛 [jQuery 初级面试题]

jQuery 第一阶段——绑定事件不再烦恼。跟着表单走，无名变有名。jQuery 选择器让你眼花缭乱，绑定事件使你出其不意就可以将对方绑定，让对方听从你的一切派遣。

13.1 要想简单，那就简单（选择器）

第 545 道：下面哪种不是 jQuery 的选择器？（ ）

A. 基本选择器 B. 后代选择器 C. 类选择器 D. 进一步选择器

答案：C

解析：

jQuery 的选择器有基本选择器、后代选择器、进一步选择器、层次选择器、基本过滤器、内容过滤器、可见性过滤器、属性过滤器、子元素过滤器、表单选择器、表单过滤器。

第 546 道：下面哪一个是用来追加到指定元素的末尾的？（ ）

A. insertAfter() B. append() C. appendTo() D. after()

答案：C

解析：

jQuery 的核心函数 appendTo() 方法是用来在被选元素的结尾（仍然在内部）插入指定内容的。

第 547 道：下面哪一个不是 jQuery 对象访问的方法？（ ）

A. each(callback) B. size() C. index(subject) D. index()

答案：D

解析：

jQuery 的核心函数之对象访问，访问对象的方法有 each(callback)、size()、index(subject)。

第 548 道：如果需要匹配包含文本的元素，用下面哪种来实现？（ ）

A. text() B. contains() C. input() D. attr(name)

答案：B

解析：

contains 选择器选取包含指定字符串的元素。该字符串可以是直接包含在元素中的文本，也可以被包含于子元素中。经常与其他元素或选择器一起使用，来选择指定的组中包含指定文本的元素。

第 549 道：用下面哪个方法可以快速找到一个表格的指定行数的元素？
()

A. text() B. get() C. eq() D. contents()

答案：C

解析：

eq() 选择器选取带有指定 index 值的元素。所以根据行数，可以很快地找到指定的元素。

第 550 道：jQuery 常见选择器有哪些？

解析：

1. 基本元素选择器

\$("#p")：选取所有 p 元素；

\$("#p.ii")：选取所有 class 是 ii 的 p 元素；

\$("#p#demo")：选取 id=demo 的第一个 p 元素。

2. 分层选择器

\$("#div input")：DIV 下的所有 input 后代元素；

\$("#div>input")：DIV 下的所有 input 子元素。

3. 基本条件选择器

\$("#p:first")：选取文档中第 1 个 p 元素；

\$("#p:last")：选取文档中最后 1 个 p 元素；

\$("#tr:even")：选择偶数行；

\$("#tr:odd")：选择奇数行；

\$("#input:not(:checked)")：选择所有未被选中的元素；

\$("#tr:eq(1)")：索引值为 1 的表格，即第二行；

\$("#tr:gt(0)")：大于 0 的行数；

\$("#tr:lt(0)")：小于 0 的行数；

\$("#:header")：选择所有标题元素 (h1 ~ h6)；

\$("#:animated")：正在执行的动画。

4. 内容条件选择器

\$("#div:constains('ddd')")：选择包含 ddd 文本的层元素；

\$("#td:empty")：选择不包含文本或者子元素的表格单元；

\$("#div:has(p)")：选择包含段落元素的层元素；

\$("#td:parent")：选择包含子元素或者文本的表格单元。

5. 可见性条件选择器

`$("tr:hidden")`: 选择所有隐藏的表格;

`$("tr:visible")`: 选择所有可见的表格。

6. 属性选择器

`$("div[id]")`: 具有 id 属性的元素;

`$("input[name=]")`: input 属性 name 是 " 的元素;

`$("input[name!=]")`: input 属性 name 不是 " 的元素;

`$("input[name^=]")`: 选择具有 name 属性并且值以 " 为起始内容的表单输入元素;

`$("input[name$=]")`: 选择具有 name 属性并且值以 " 为结束内容的表单输入元素;

`$("input[name*=]")`: 选择具有 name 属性并且值以 " 的表单输入元素;

`$("input[id][name$=]")`: 选择具有 id 和 name 属性的值以 " 为结束内容的输入表单元素。

7. 子元素选择器

`$("ul li:nth-child(2)")`: 选择第二个列表项;

`$("ul li:nth-child(even)")`: 表示匹配作为父元素的偶数 (第 2、4、6、8...) 个子元素的元素;

`$("ul li:nth-child(odd)")`: 表示匹配作为父元素的奇数 (第 1、3、5、7...) 个子元素的元素;

`$("ul li:nth-child(3n)")`: 表示匹配作为父元素的第 3n 个元素 (n 表示包括 0 在内的自然数, 下同);

`$("ul li:first-child")`: 表示匹配父元素的第 1 个元素;

`$("ul li:last-child")`: 表示匹配父元素的最后 1 个元素;

`$("ul li:only-child")`: 选择列表出现且仅出现一个列表项。

8. 表单元素选择器

`$("input")`: 选择所有 input、textarea、select、button 等元素;

`$(":text")`: 文本行;

`$(":password")`;

`$(":radio")`;

`$(":checkbox")`;

`$(":submit")`;

`$(":image")`;

`$(":reset")`;

`$(":button")`;

`$(":file")`;

`$(":hidden")`。

9. 表单属性选择器

`$("input:enabled")`: 选择所有可用;

`$("input:disabled")`;

`$("input:checked")`;

`$("select:option:selected")`。

第 551 道: jQuery 中的选择器和 CSS 中的选择器有区别吗?

解析:

jQuery 有 9 种选择器，分别是基础选择器和层级选择器，它与 CSS 中的选择器有些类似，jQuery 中的选择器所获得的是 DOM (Document Object Model) 对象，而 CSS 中的选择器仅仅是给标签增加样式。

第 552 道: jQuery 中的选择器有什么优势?

解析:

(1) 简洁的写法，`$()` 函数;

(2) 支持 CSS1 到 CSS3 选择器;

(3) 完善的处理机制。

第 553 道: 在 jQuery 选择器中，id 与 class 有什么区别?

解析:

class 选择器获取到的是一类对象的集合，id 选择器是获取一个对象，就好比姓张的人可能有 N 个，而身份证为 XXXXXXXXXXXX 的人只有一个。

第 554 道: `siblings()` 方法和 `$('prev~div')` 选择器是一样的吗?

解析:

`siblings()` 是获得匹配集合中每个元素的同胞，而 `$('prev~div')` 是获得当前元素之前的匹配集合中每个元素的同胞。

第 555 道: 请用 jQuery 实现在 body 中插入一个 `class="box"` 的 DIV。

解析:

```
$('body').append($('div.box'));
```

13.2 面试总会有不期而遇的温暖（属性）

第 556 道: 如果想在指定的元素后添加内容，下面哪个可以实现该功能 ()

A. `append(content)`

B. `appendTo(content)`

C. `insertAfter(content)`

D. `after(content)`

答案: D

解析:

append(content)——向元素的末尾添加 HTML 代码;

appendTo(content)——添加 HTML 代码到元素的末尾;

insertAfter(content)——将 jQuery 对象插入指定元素的后面;

after(content)——将 HTML 代码插入指定元素的后面。

第 557 道: 在 jQuery 中, 想要给第一个指定的元素添加样式, 下面哪一个是正确的? ()

A. first B. eq(1) C. css(name) D. css(name,value)

答案: C

解析:

css(name) 用来获取第一个匹配元素的样式属性, 并且 css(name) 不仅可以获取使用 css 定义的属性中的值, 同样可以获取 HTML 标记中的属性值。

第 558 道: 在 jQuery 中, 如果想要获取当前窗口的宽度值, 下面哪个可以实现该功能? ()

A. width() B. width(val) C. width D. innerWidth()

答案: D

第 559 道: 在一个表单中, 如果想要给输入框添加一个输入验证, 可以用下面哪个事件实现? ()

A. hover(over,out) B. keypress(fn) C. change() D. change(fn)

答案: B

解析:

keypress 事件与 keydown 事件类似。当按钮被按下时, 会发生该事件。它发生在当前获得焦点的元素上。不过, 与 keydown 事件不同, 每插入一个字符, 就会发生 keypress 事件。当键盘被按下并松开时, 触发 keypress 事件, 它用来做输入框验证。

第 560 道: 当一个文本框中的内容被选中时, 想要执行指定的方法, 可以使用下面哪个事件来实现? ()

A. click(fn) B. change(fn) C. select(fn) D. bind(fn)

答案: C

解析:

select() 的作用是在每一个匹配元素的 select 事件中绑定一个处理函数。当用户在文本框 (包括 input 和 textarea) 中选中某段文本时会触发 select 事件。

第 561 道: 在 jQuery 中想要实现通过远程 HTTP GET 请求载入信息功能可以使用下面哪个事件来实现? ()

A. \$.ajax() B. load(url) C. \$.get(url) D. \$.getScript(url)

答案: C

解析：

get() 方法通过远程 HTTP GET 请求载入信息。这是一个简单的 GET 请求功能，用以取代复杂 \$.Ajax 请求成功时可调用的回调函数。

第 562 道：在 jQuery 中想要找到所有元素的同辈元素，下面哪一个是可以实现的（ ）

A. eq(index) B. find(expr) C. siblings([expr]) D. next()

答案：C

解析：

siblings() 获取每个匹配元素的前后所有的同辈元素，然后作为一个 jQuery 包装集返回。另外，还可以传递一个选择器表达式参数给它，并在这些同辈元素中进行筛选。

第 563 道：在 jQuery 中，用一个表达式来检查当前选择的元素集合，使用 _____ 来实现，如果这个表达式失效，则返回 _____ 值。

答案：is, false

解析：

如果没有元素符合，或者表达式无效，都返回 “false”，“filter” 内部也在调用 is() 这个函数，所以，filter() 函数原有的规则在这里也适用。

第 564 道：彻底将 jQuery 变量还原，可以使用 _____ 方法实现。

答案：noConflict() 方法

解析：

(1) noConflict() 方法让渡变量 \$ 的 jQuery 控制权。

(2) 该方法释放 jQuery 对 \$ 变量的控制。

(3) 该方法也可用于为 jQuery 变量规定新的自定义名称。

第 565 道：子元素选择器和后代元素选择器有什么区别？

解析：

子元素选择器只对直接后代有影响，而对 “孙子后代” 以及多层后代不产生作用。而后代元素选择器对 “孙子后代” 以及多层后代都产生作用。

比如说：一家人祖孙三代，爷爷、爸爸、叔叔、孙子。爷爷的子元素是父亲和叔叔，爷爷的后代元素是爸爸、叔叔、孙子。

13.3 不要为面试而烦恼（绑定事件）

第 566 道：为每一个指定元素的指定事件（像 click）绑定一个事件处理器函数，下面哪个是用来实现该功能的？（ ）

A. trigger(type) B. bind(type)

C. one(type)

D. bind

答案: B

解析:

bind(type) 方法是用于往文档上附加行为的主要方式。所有 JavaScript 事件对象, 比如 focus、mouseover 和 resize, 都可以作为 type 参数传递进来。

第 567 道: 下面哪几个不是属于 jQuery 的事件处理 () (多选)

A. bind(type)

B. click()

C. change()

D. one(type)

答案: BC

解析:

只有 bind(type) 和 one(type) 属于事件处理, 而 click() 和 change() 不属于事件处理。

第 568 道: 在 jQuery 中指定一个类, 如果存在就执行删除功能, 如果不存在就执行添加功能, 下面哪一个是可以直接完成该功能的? ()

A. removeClass()

B. deleteClass()

C. toggleClass(class)

D. addClass()

答案: C

解析:

toggleClass() 对设置或移除被选元素的一个或多个类进行切换。该方法检查每个元素中指定的类。如果不存在, 则添加类; 如果已设置, 则删除之。这就是所谓的切换效果。

第 569 道: 请写出不少于两种 jQuery 绑定事件的方法。

解析:

有 on()、bind()、live()、delegate() 等方法。前两个方法对后创建的元素不起作用, 后两个方法对后创建的元素起作用。

13.4 不要害怕面试, 因为你需要 (表单)

第 570 道: jQuery 表单提交前有几种校验方法? 分别是什么

解析:

有三种校验方法, 分别如下:

(1) formData: 返回一个数组, 可以通过循环调用来校验。

(2) jaForm: 返回一个 jQuery 对象, 所有需要先转换成 DOM 对象。

(3) flEldValue: 返回一个数组。

13.5 说多了都是眼泪，还是来点实在的吧（文档处理）

第 571 道：下面哪几种属于 jQuery 文档处理（ ）（多选）

A. 包裹 B. 替换 C. 删除 D. 内部和外部插入

答案：ABCD

解析：

在使用 jQuery 进行文档处理时，总共分为 6 种模式：内部插入、外部插入、包裹、替换、删除、复制。

13.6 最怕的东西，最应该去突破（筛选）

第 572 道：下面哪种不属于 jQuery 的筛选（ ）

A. 过滤 B. 自动 C. 查找 D. 串联

答案：B

解析：

jQuery 的筛选函数提供了串联、查找和过滤函数，是不包括自动的。

第 14 章



学贯古今，中流砥柱 [jQuery 中级面试题]

jQuery 第二阶段——还在裸奔面试吗？有 DOM 加载，你就是冷面杀手；移动端事件帮你解决各种修炼问题，在将来的江湖道路上必将祝你一臂之力。

14.1 你还在“泡”招聘，“奔”面试吗（DOM 加载）

第 573 道：当 DOM 加载完成后要执行的函数，下面哪个是正确的（ ）

- A. jQuery (expression, [context])
- B. jQuery(html,[ownerDocument])
- C. jQuery(callback)
- D. jQuery(elements)

答案：C

解析：

jQuery(callback) 函数在 DOM 加载完成之后执行。

第 574 道：在 jQuery 中，如果想要从 DOM 中删除所有匹配的元素，下面哪一个是正确的（ ）

- A. delete()
- B. empty()
- C. remove()
- D. removeAll()

答案：D

解析：

removeAll() 用来从 DOM 中彻底删除所有匹配的元素。

14.2 你能让面试官惊呆吗（移动端事件）

第 575 道：touches、targetTouches、changedTouches 三个属性有什么区别？

解析：

- (1) touches：当前位于屏幕上的所有手指的一个列表。
- (2) targetTouches：位于当前 DOM 元素上的手指的一个列表。
- (3) changedTouches：涉及当前事件的手指的一个列表。

14.3 面试是一张网，你收获了吗（取 HTML、文本的值）

第 576 道：如何来设置和获取 HTML 和文本的值？

解析：

可通过 `html()` 及 `text()` 方法获取并设置 HTML 和文本的值。

第 577 道：`$("#msg").text();` 和 `$("#msg").text("new content");` 有什么区别？

解析：

`$("#msg").text()` 是获取当前选择器的 HTML 和文本的值，而 `$("#msg").text("new content")` 是给当前选择器设置 HTML 和文本的值。

第 578 道：jQuery 怎么判断元素是否存在？

解析：

在 jQuery 中，我们可使用 `$("#div").length > 0` 来判断元素是否存在，意思就是判断元素长度，如果没有肯定是不存在的。

14.4 每天超越自己一点点（事件）

第 579 道：jQuery 中，`$.bind()`、`$.live()` 和 `$.delegate()` 的区别是什么？

解析：

`bind()` 方法是为被选元素添加一个或多个事件处理程序，并规定事件发生时运行的函数。将事件和函数绑定到元素，规定向被选元素添加的一个或多个事件处理程序，以及当事件发生时运行的函数。

`live()` 方法是为被选元素附加一个或多个事件处理程序，并规定当这些事件发生时运行的函数。

`delegate()` 方法将方法绑定到特定的元素，而将 `live()` 绑定到 document 根元素。

第 580 道：什么是事件委派？jQuery 用事件委派怎么写？

解析：

事件委派：一次性绑定多个事件，根据事件类别来委派相应的操作。

写法：

`$("#select").bind("mouseover",function(){}).bind("mouseout",function(){});`

第 581 道：在移动 Web 开发时，可以使用 click 事件吗？如不行，请说明原因？

解析：

在移动端使用 click 事件有可能会出现点透的情况，即单击会触发非当

前层的单击事件。单击事件可分解为多个事件，手指单击会经过：touchstart->touchmove->touchend->click。

所以说在移动端尽可能不使用 click 事件，因为 click 事件可能会触发其他的事件。

第 582 道：写出你所知道 jQuery 事件的部分方法。

解析：

blur() 元素上失去焦点：a,input,textarea,button,select,label.map,area;

change() 用户改变域的内容：input,textarea,select;

click() 鼠标单击某个对象；

dblclick 鼠标双击某个对象；

Error() 当加载文档或图像时发生某个错误；

Focus() 元素获焦点：a,input,textarea,button,select,label.map,area;

keydown() 键盘的某个键被按下；

keypress () 键盘的某个键被按下或按住；

keyup() 键盘的某个键被松开；

mousedown (fn) 鼠标某个按键被按下；

mousemove (fn) 鼠标被移动；

mouseout (fn) 鼠标从某个元素离开。

第 583 道：请简述 jQuery 中 \$.live() 的实现原理。

解析：

其实就是将事件绑定到父节点，由于事件冒泡，所有事件最终会冒泡到 document 节点。当有事件触发时，则判断事件类型和触发事件的元素是否一致，如果相同则执行函数。

第 584 道：jQuery 中 proxy 及 delegate 两个方法的作用是什么？

解析：

1. proxy 方法：通俗地讲就是改变函数的作用域，有两种调用方式：

(1) jquery.proxy(function,context)

function 为将要执行的函数，context 为 function 上下文对象。

(2) jquery.proxy(context,name)

context 为函数上下文 object 对象，name 为将要执行的函数。

2. delegate 方法：为匹配到的选择器绑定一个或者多个事件。调用方式：

delegate(selector,eventType,handler);

selector 为选择器，eventType 为事件类型（包含一个或者多个事件的字符串），handler 为事件触发执行的函数。

第 15 章



出神入化，学贯古今 [jQuery 高级面试题]

jQuery 第三阶段——read、onload 来者不拒，实实在在的千人斩。

15.1 做小题，成大事（read、onload 的区别）

第 585 道：window.onload 和 document.ready() 的区别？

解析：

window.onload 方法是在网页中的所有的元素（包括元素的所有关联文件）都完全加载到浏览器之后才执行。这种方式有一个很大的优点：不用考虑 DOM 元素加载的顺序。

而通过 jQuery 中的 \$(document).ready() 方法注册的事件处理程序，只要在 DOM 完全就绪时，就可以调用了，比如一张图片只要标签完成，不用等这个图片加载完成，就可以设置图片宽、高的属性或样式等。这种方式优于 onload() 事件的原因在于：\$(document).ready() 可以在页面没有完全下载时，操作页面的 DOM 元素。

第 586 道：jQuery 中的 load 方法怎么用？

解析：

```
load(url,[data],[callback]);
```

url：要导入文件的地址；

data：可选参数，因为 load 不仅仅可以导入静态的 HTML 文件，还可以导入动态脚本，例如 PHP 文件，所以要导入动态文件时，我们可以将传递的参数放在里面；

callback：可选参数，是指调用 load 方法并得到服务器响应后，再执行的另一个参数。

15.2 让愤怒多些实力（效果）

第 587 道：在一个表单中，用 600ms 缓慢地将段落滑上，用 _____

来实现。

答案: `$("p").hide("slow");`

解析:

`hide()` 方法是将显示的元素隐藏。

第 588 道: 在 jQuery 中, 如果想要自定义一个动画, 用 _____ 函数来实现。

解析:

`animate(params,options,function(){})`

第 589 道: 在 jQuery 中, 想让一个元素隐藏, 用 _____ 实现, 显示隐藏的元素用 _____ 实现。

解析:

让一个元素隐藏可以用 `hide()` 方法和 `display:none;`。

让一个元素显示可以用 `show()` 方法和 `display:block;`。

第 590 道: 如何使用 jQuery 实现复选框的全选和反全选效果?

解析:

```
<script type="text/JavaScript">
    /* 全选 */
    /* items 复选框的 name */
    function allCkb(items){
        $(' [name='+items+']:checkbox').attr("checked", true);
    }
    /* 全不选 */
    function unAllCkb(){
        $(' [type=checkbox]:checkbox').attr('checked', false);
    }
    /* 反选 */
    /*items 复选框的 name */
    function inverseCkb(items){
        $(' [name='+items+']:checkbox').each(function(){
            // 此处用 jQuery 写法颇显啰嗦。体现不出 jQuery 飘逸的感觉
            // $(this).attr("checked", !$ (this).attr("checked"));
            // 直接使用 JS 原生代码, 简单实用
            this.checked=!this.checked;
        });
    }
</script>
```

第 591 道: 如何用 jQuery 实现淡入淡出的效果?

解析:

1. jQuery `fadeIn` 方法用于淡入已隐藏的元素。

`$(selector).fadeIn(speed,callback);`

`speed` (可选) 参数规定效果的时长, 它可以选取以下参数值: "slow", "fast" 或毫秒。

`callback` 参数是 `fadeIn()` 完成后所执行的函数名称。

2. jQuery fadeOut 方法用于淡出可见的元素。

`$(selector).fadeOut(speed,callback);`

speed（可选）参数规定效果的时长，它可以选取以下参数值："slow"，"fast" 或毫秒。

callback 参数是 fadeOut() 完成后所执行的函数名称。

3. jQuery fadeToggle() 方法可以在 fadeIn() 与 fadeOut() 方法之间进行切换。

如果元素已淡出，则 fadeToggle() 会向元素添加淡入效果；

如果元素已淡入，则 fadeToggle() 会向元素添加淡出效果；

`$(selector).fadeToggle(speed,callback);`

speed（可选）参数规定效果的时长，它可以选取以下参数值："slow"，"fast" 或毫秒。

callback 参数是 fadeToggle() 完成后所执行的函数名称。

4. jQuery fadeTo() 方法允许渐变为给定的不透明度（值介于 0 到 1 之间）。

`$(selector).fadeTo(speed,opacity,callback);`

speed（必须）参数规定效果的时长，它可以选取以下值："slow"，"fast" 或毫秒。

fadeTo() 方法中 opacity 的参数将淡入淡出效果设置为给定的不透明度（值介于 0 到 1 之间）。

callback 参数是 fadeTo() 完成后所执行的函数名称。

第 592 道：如何使用 jQuery 实现当单击按钮时弹出一个对话框？

解析：

```
$("div").click(function(){
    Alert(“单击我！”);
})
```

第 16 章



英姿勃发，独当一面 [jQuery 终极面试题]

jQuery 第四阶段——Ajax 在身旁，你就是一派宗师。get、post 两种获取方式将给你带来无限精彩，jQuery、DOM 对象更是永垂不朽。

16.1 困难的考题能让你看到更多的风景（get 和 post）

第 593 道：请指出 eq(),get(),[] 的区别。

解析：

eq 返回的是一个 jQuery 对象；

get 返回的是一个 HTML 对象数组；

var array=[] 声明一个数组。

16.2 想知道你能力的边界在哪吗（优化）

第 594 道：介绍一下 jQuery easyUI。如何使用 jQuery easyUI 组件？

解析：

jQuery EasyUI 是一组基于 jQuery 的 UI 插件集合体，而 jQuery EasyUI 的目标就是帮助 Web 开发者更轻松地打造出功能丰富并且美观的 UI 界面。开发者不需要编写复杂的 JavaScript，也不需要深入了解 CSS 样式，开发者需要了解的只有一些简单的 HTML 标签。

首先 easyUI 的 datagrid 插件，返回的数据可以是 JSON，结构为 {"total":239,"rows":[{"code":"001","name":"Name1","addr":"Address11","col4":"col4data"}, {"code":"002","name":"Name2","addr":"Address13","col4":"col4 data"}]}。

除了 total 和 rows，还可以有其他项，但是这两项必须有。而在对应的后台数据可以是 map，也可以是 JSON。当是 map 时，同样需要在 map 中放入 total、rows 这两个 key。

第 595 道：你知道哪些针对 jQuery 的优化方法？

解析：

- (1) jQuery 代码重构。
- (2) 尽可能设置全局变量。
- (3) 避免出现重复的代码，用尽可能少的代码实现效果。

第 596 道：以下 jQuery 的常用操作是如何实现的？

- (1) 获取属性为 username 的 input 值。
- (2) 获取 ul 下第 n 个 li 的文本。
- (3) 将 class="latest" 的元素插入 id="list" 的最前面。
- (4) 简述在 jQuery 下 Ajax 的过程。

解析：

```
(1) attr("username").val()。
(2) $("ul li").eq(n)。
(3) $(".latest").prepend To($("#list"))。
(4) $.ajax( {
  type : "post",
  dataType : "JSON",
  url : "<%=basePath%>/prod/findTypeName.do",
  async : true,
  success : function(result){
    alert(" 成功 ");
  },
  error:function(){alert(" 失败 ")}
});
```

16.3 其实成功一直在你的旁边 (Ajax)

第 597 道：下面不属于 Ajax 事件的是 ()

- A. ajaxComplete(callback)
- B. ajaxSuccess(callback)
- C. \$. post(url)
- D. ajaxSend(callback)

答案：C

解析：

1. ajaxComplete(callback)

Ajax 请求完成时执行函数，属于 Ajax 事件。

参数：callback 代执行函数。

例如：

```
$("#msg").AjaxComplete(function(event,request,settings){
  $(this).append("<li> 请求完成 </li>")})
```

2. ajaxSuccess(callback)

Ajax 请求顺利完成时执行函数，属于 Ajax 事件。

返回值：jQuery。

例如 jQuery 代码：

```
$("#msg").AjaxSuccess(
    function(request, settings){
        $(this).append("<li>Successful Request!</li>");
    });
```

3. ajaxSend(callback)

Ajax 请求发送前执行函数，属于 Ajax 事件。

XMLHttpRequest 对象和设置作为参数传递给回调函数。

Ajax 请求发送前显示信息。

例如 jQuery 代码：

```
$("#msg").AjaxSuccess(
    function(request, settings){
        $(this).append("<li> 开始请求: "+setting.url+ "</li>");});
$.post(url);
$.post(url, data, callback);
```

- url 参数必须规定您希望请求的地址；
- data 参数可选规定连同请求发送的数据；
- callback 参数可选是请求成功后执行的函数名。

16.4 放手做，勇敢错（jQuery、DOM 对象）

第 598 道：你知道哪些不好的 jQuery 书写方式。

解析：

没有加 \$ 符号；没有加选择器；选择器中没有加引号。

第 599 道：在 jQuery 中，请给出下面代码的执行结果。

```
<b>Hello</b> <p> Sogou</p>
$( "b" ).clone().prependTo( "p" );
```

解析：

\$("b").clone().prependTo("p"); 的结果是 Hello,Sogou;
prependTo() 方法将元素添加到指定元素之前。

第 600 道：下面哪一个不是 jQuery 对象访问的方法？

- A. each(callback) B. size() C. index(subject) D. index()

答案：C

解析：

each(callback): 以每一个匹配的元素作为上下文来执行一个函数。它意味着每次执行传递进来的函数时，函数中的 **this** 关键字都指向一个不同的 DOM 元素（每次都是一个不同的匹配元素）。而且，在每次执行函数时，都会给函数传递一个表示作为执行环境的元素在匹配的元素集合中所处位置的数字值作为参数（从零开始的整型）。返回 **false** 将停止循环（就像在普通的循环中使用 **break**）；返回 **true** 跳至下一个循环（就像在普通的循环中使用 **continue**）。**callback**：每个匹配的元素所要执行的函数。

size(): jQuery 对象中元素的个数。这个函数的返回值与 jQuery 对象的 **length** 属性一致。返回值为 **Number**。

index(): 返回指定元素相对于其他指定元素的 **index** 位置，这些元素可通过 jQuery 选择器或 DOM 元素来指定。

第 601 道：jQuery 插件实现方法有哪些？请分别介绍。

解析：

1. 添加静态方法

`jQuery.extend(object);`

例如：

```
$.extend({
  addMethod : function(a, b){return a + b;}
  // $.addMethod(1, 2); //return});
```

2. 添加成员方法

`jQuery.fn.extend(object);`

例如：

```
jQuery.fn = jQuery.prototype;
```

给 jQuery 对象添加方法，对 `jQuery.prototype` 进行扩展，为 jQuery 类添加成员方法：

```
$.fn.extend({
  getInputText:function(){
    $(this).click(function(){
      alert($(this).val());
    });
  }
});
$("#username").getInputText();
```

第 602 道：DOM 对象如何转成 jQuery 对象？

解析：

如果已经是一个 DOM 对象，那么只需要用 `$()` 把 DOM 对象包装起来，就可以获得一个 jQuery 对象了。即 `$(DOM 对象)`。

如：

```
var v=document.getElementById("v"); //DOM 对象
var $v=$(v); //jQuery 对象
```

转换后，就可以任意使用 jQuery 方法了。

第 603 道：jQuery 对象如何转成 DOM 对象？

解析：

有两种转换方式，可以将一个 jQuery 对象转换成 DOM 对象，即：`[index]` 和 `get(index)`。jQuery 本身，通过 `.get(index)` 方法，得到相应的 DOM 对象。

jQuery 对象是一个数据对象，可以通过 `[index]` 的方法来得到相应的 DOM 对象。

(1) `[index]` 方法

例如：

```
var $v = $("#v") ; //jQuery 对象
var v=$v[0]; //DOM 对象
alert(v.checked) // 检测这个 checkbox 是否被选中
```

(2) `get(index)` 方法

例如：

```
var $v=$("#v"); //jQuery 对象
var v=$v.get(0); //DOM 对象
alert(v.checked) // 检测这个 checkbox 是否被选中
```

第 604 道：用 jQuery 实现下面的登录表单验证。

要求：

- (1) 用户为空时，提示：用户名不能为空。
- (2) 密码为空时，提示：密码不能为空。
- (3) 回车可以验证并提交表单。
- (4) 使用 post 方式提交表单。
- (5) 提示都显示在 `p.prompt` 中。

解析：

```
<script type="text/JavaScript">
    function yanzheng(){
        if($("#name").val() == ""){
            alert (" 用户名不能为空！");
            p.prompt(" 用户名不能为空！");
        } else if($("#password").val() == ""){
            alert (" 密码不能为空！");
            p.prompt(" 密码不能为空！");
        }
    }
    document.onkeydown = function(event){
        if(event.keyCode==13){
            yanzheng();
        }
    }
    $("#tijiao").click(function(){
        yanzheng();
    })
</script>
```


第 605 道：你为什么要使用 jQuery ？

解析：

jQuery 是继 prototype 之后又一个优秀的 JavaScript 框架。其宗旨是写更少的代码，做更多的事情。

它是轻量级的 JS 库（压缩后只有 21kB），这是其他的 JS 库所不及的，它兼容 CSS3，还兼容各种浏览器（IE 6.0+、Firefox 1.5+、Safari 2.0+、Opera 9.0+）。

jQuery 是一个快速的、简洁的 JavaScript 库，它使用户能更方便地处理 HTML 文档遍历、事件处理，实现动画效果，并且能方便地为网站提供 Ajax 交互。

jQuery 还有一个比较大的优势是，它的文档说明很全，而且各种应用也说得 very 详细，同时还有许多成熟的插件可供选择。

jQuery 能够使用户的 HTML 页面保持代码和内容分离，也就是说，不用在 HTML 里面插入一堆 JS 来调用命令，只需定义 id 即可。

16.5 断了退路，才有出路（\$.getScript() 和 \$.getJSON()）

第 606 道：\$.getScript() 方法 和 \$.getJSON() 方法有什么区别？

解析：

\$.getScript() 方法通过 HTTP GET 请求载入并执行 JavaScript 文件。

语法：\$.getScript(url, success(responses, status))。

(1) url：将要请求的 url 字符串。

(2) success(response,status)：可选。规定请求成功后执行的回调函数。

额外的参数：

.response——包含来自请求的结果数据；

.status——包含请求的状态（"success", "notmodified", "error", "timeout"

或 "parsererror"）。

\$.getJSON() 通过 HTTP GET 请求载入 JSON 数据。

语法：jQuery.getJson(url,data,success(data, status,xhr));

(1) url：必需。规定将请求发送给哪个 url。

(2) data：可选。规定连同请求发送到服务器的数据。

(3) success(data,status,xhr)：可选。规定当请求成功时运行的函数。

额外的参数：

.date —— 包含来自请求的结果数据；

.status —— 包含请求的状态；

.xhr —— 包含 XMLHttpRequest 对象。

第 607 道：为什么要使用第三方工具 CDN 来加速 jQuery 文件？

解析：

- (1) 减少等待时间。
- (2) 增加网页的同时载入速度。
- (3) 更好的缓存。

第 608 道：jQuery 中哪些方法不能在 zepto 中使用，举一个例子即可。

解析：

例如：在 jQuery 中有 fadeIn 和 fadeOut，用来实现渐隐渐显的效果，而在 zepto 中没有 fadeIn 和 fadeOut，但是在 zepto 中我们可以通过动画效果 animate 来实现渐隐渐显的效果。

第 609 道：jQuery 是否支持 AMD 规范？当作为 AMD 模块运行时，模块名是什么？

解析：

AMD 模块，AMD（Asynchronous Module Definition，异步模块定义）格式总体的目标是为现在的开发者提供一个可用的模块化 JavaScript 的解决方案。

AMD 模块格式本身是一个关于如何定义模块的提案，在这种定义下模块和依赖项都能够异步地进行加载。它有很多独特的优势，包括天生的异步及高度灵活等特性，这些特性能够解除常见的代码与模块标识间的那种紧密耦合。目前它已经被很多项目所接纳，包括 jQuery。

第 610 道：JS 如何实现循环复制一个 DIV。

解析：

```
position:absolute; <html>
<head>
<title>Test of cloneNode Method</title>
<script type="text/JavaScript" src="test.JS"></script>
</head>
<body>
<div id="main">
<div id="div-0">
<span>Cloud018 said, </span>
<span>"Hello World!!!"</span>
</div>
</div>
</body>
</html>

// test.JS
```

```

window.onload = function () {
    var sourceNode = document.getElementById("div-0"); // 获得被
克隆的节点对象
    for (var i = 1; i < 5; i++) {
        var clonedNode = sourceNode.cloneNode(true); // 克隆节点
        clonedNode.setAttribute("id", "div-" + i); // 修改一下 id 值，
避免 id 重复
        sourceNode.parentNode.appendChild(clonedNode); // 在父节点插
入克隆的节点
    }
}

```

第 611 道：jQuery 的美元符号 \$ 有什么作用？

解析：

其实美元符号 \$ 只是“jQuery”的别名，它是 jQuery 的选择器，代码如下：

```

$(document).ready(function() {
});

```

当然，你也可以用 jQuery 来代替 \$，代码如下：

```

jQuery(document).ready(function() {
});

```

在 jQuery 中，就是通过这个美元符号来实现各种灵活的 DOM 元素选择的，例如 \$("#main") 即选中 id 为 main 的元素。

第 612 道：请使用 jQuery 将页面上的所有元素边框设置为 2px 的虚线。

解析：

这正是该使用 jQuery 选择器的时候了，代码如下：

```

<script language="JavaScript" type="text/JavaScript">
    $("*").css("border", "2px dotted red");
</script>

```

第 613 道：怎样用 jQuery 实现对 URL 的编码和解码？

解析：

在 jQuery 中，我们可以使用以下方法实现 URL 的编码和解码：

encodeURIComponent(url) 和 decodeURIComponent(url)。

第 614 道：如何用 jQuery 禁用浏览器的前进后退按钮？

解析：

实现代码如下：

```

<script type="text/JavaScript" language="JavaScript">
$(document).ready(function() {
    window.history.forward(1);
    //OK
    window.history.forward(-1);
});

```

```
});
</script>
```

第 615 道：如何解决 Query、prototype 共存，以及 \$ 全局变量冲突问题？

解析：

- (1) 将 jQuery.JS 放到 prototype.JS 后面（这个是必须的）。
- (2) 在 jQuery.JS 后面将 \$ 变量重命名。

代码如下：

```
<script type="text/javascript" src="window.JS"></script>
<!-- 上面这个 window.JS 调用了 jQuery 框架的方法 -->
<script type="text/javascript" src="prototype.JS"></script>
<script type="text/javascript" src="jQuery.JS"></script>
<script type="text/javascript">
var jQuery=$;
</script>
```

- (3) 将原来使用的 \$ 方法名统一替换为 jQuery 名，如 \$("obj") 替换为 jQuery("obj")。

第 616 道：如何使用 jQuery 来切换样式表？

解析：

找出你希望切换的媒体类型（media-type），然后把 href 设置成新的样式表。

`$('link[media='screen']").attr('href', 'Alternative.css');`

第 617 道：如何正确地使用 toggleClass ？

解析：

切换（toggle）类允许你根据某个类是否存在来添加或是删除该类。在这种情况下有些开发者使用：

```
a.hasClass('blueButton');
a.removeClass('blueButton');
a.addClass('blueButton');
```

toggleClass 允许你使用下面的语句，这样就很容易地做到这一点。

```
a.toggleClass('blueButton');
```

第 618 道：写一段代码描述命令模式。

解析：

```
var command=function(receiverObj){
    this.receiver=receiverObj;
}
command.prototype={
    exec:function(){
        this.receiver.doSomething();// 接收者的具体操作
    },
};
```

```
undo:function(){
    this.receiver.backout();//撤销操作
}
};
var invoker=function(cmdObj){
    this.command=cmdObj;
}
invoker.prototype={
    handle:function(){//调用者的一个操作
        this.command.exec();
        //不需要知道操作的对象是谁，有接收者负责，仅执行操作即可，
        //这样可以随意更改command参数，只要它实现了exec方法；
    }
};
```

第619道：jQuery 能实现什么效果？

解析：

动画效果、左右滑动效果、淡入淡出效果、显示隐藏效果、实现全选效果、遮罩效果、loading 效果。

第5篇

「17助力」泯恩仇， 清风落，江湖生

“17助力”结合多种关键要素，解决了在开发过程中的一些难点难题，例如浏览器兼容问题。并且还为大家准备了一些利于使用而且操作简单的框架、插件和一些JS库，使大家在开发过程中更为简单方便，为大家在开发过程中添加了左膀右臂。

第 17 章



万事俱备，只欠东风 [“17助力”，助你一臂之力]

“17 助力”多才多艺，命运把握在自己手上，成功唾手可得，更是在未来的江湖道路上让你登峰造极。

17.1 “助力 1”：浏览器和兼容差异

第 620 道：请列举 JavaScript 在 IE 和 W3C 系列浏览器中的兼容差异。

解析：

浏览器中的兼容问题是非常常见的，JavaScript 有以下几个兼容差异问题：

- (1) 函数和方法差异。
- (2) 样式访问和设置。
- (3) DOM 方法及对象引用。
- (4) 事件处理。

第 621 道：IE 与 Firefox 的 JS 兼容性都有什么？

解析：

- (1) document.formName.item 问题。
- (2) 集合对象的问题。
- (3) window.event 的问题。
- (4) HTML 对象的 id 作为对象名的问题。
- (5) 用 idName 字符串取得对象的问题。
- (6) 变量名与某 HTML 对象 id 相同的问题。
- (7) event.x 与 event.y 的问题。
- (8) frame 的问题。
- (9) const 的问题。
- (10) body 对象的问题。
- (11) nodeName 和 tagName 的问题。
- (12) 元素属性。
- (13) document.getElementsByName() 和 document.all[name] 的问题。
- (14) DOM 数据岛的问题。

第 622 道：你都知道哪些浏览器内核？

解析：

- (1) Trident 内核，代表产品 Internet Explorer。
- (2) Gecko 内核，代表产品 Mozilla Firefox。
- (3) WebKit 内核，代表产品 Safari、Chrome。
- (4) Presto 内核，代表产品 Opera。

第 623 道：你应该做了很多网页吧，那么都在哪些常见的浏览器测试过？遇到过兼容性问题吗？怎么轻易解决它们？

解析：

IE 内核浏览器：世界之窗、腾讯 TT、360、傲游、搜狗。

非 IE 内核浏览器：Chrome、Firefox、Opera、Safari。

(1) IE6 比较容易出现双倍边距的问题，在使用了 float（浮动）的情况下，不管是向左还是向右，都会出现双倍边距的问题，最简单的解决方法就是用 display: inline; 加到 CSS 里面去。

(2) 文字本身的大小不兼容。同样是 font-size: 14px 的宋体文字，在不同浏览器下占的空间是不一样的，在 IE 下实际占高 16px，下留白 3px；在 Firefox 下实际占高 17px，上留白 1px，下留白 3px；Opera 下就更不一样了。解决方法：给文字设定 line-height，确保所有文字都有默认的 line-height 值。这点很重要，在高度上我们不能容忍 1px 的差异。

(3) 在 Firefox 下容器高度限定，即容器定义了 height 之后，容器边框的外形就确定了，不会被内容撑大，而 IE 下是会被内容撑大的，高度限定失效。所以不要轻易给容器定义 height。

(4) 横向上的内容撑破容器问题。如果 float 容器未定义宽度，在 Firefox 下内容会尽可能撑开容器宽度，IE 下则会优先考虑内容折行。故内容可能撑破的浮动容器需要定义 width。

(5) 浮动的清除，在 Firefox 下不清除浮动是不行的。

(6) mirrormargin Bug，当外层元素内有 float 元素时，外层元素如定义 margin-top: 14px，将自动生成 margin-bottom: 14px。padding 也会出现类似的问题，这都是 IE6 下的“特产”，该类 Bug 出现的情况较为复杂，远不只这一种出现条件，还没系统整理。解决方案：外层元素设定 border 或设定 float。

第 624 道：浏览器模式一共分为几种？

解析：

浏览器模式分为两种：本地和 Web。

第 625 道：列举常用的浏览器类型，以及它们对应的调试工具和使用的内核。

解析：

常用的浏览器类型有 IE(6,7,8,9,10)、Firefox、Chrome。

IE 常用的调试工具有 IEWebDeveloper (IE9 默认有安装)。

大家用得最多的估计是 Firefox, 因为 Firefox 上确实有一些比较优秀的开发和调试工具, 如久负盛名的 Firebug。

chrome (内核 webkit) 自带的有 google 开发的内置调试工具。

三者内核各不相同。

其他还有 Opera、遨游、世界之窗等。

Chrome 内核跑得比较快, 安全。Firefox 做调试是最棒的。

第 626 道: 哪些属性类函数在 Firefox 中有, 但是在 Internet Explorer 中没有, 请至少写出三种。

解析:

(1) document.form.item。

(2) window.event。

(3) idName。

第 627 道: 如何清除图片下方出现的几个像素的空白间隙?

解析:

有以下方法:

(1) `img{display:block;}`。

(2) `img{vertical-align:top;}`。

(3) `# test{font-size:0;line-height:0;}`。

除了 top 值, 还可以设置为 text-top、middle bottom、text-bottom, 甚至特定的 `<length>` 和 `<percentage>` 值都可以。

第 628 道: 请列举一些 IE 与其他浏览器不一样的特性。

解析:

这个问题要分三个方面来回答:

有一部分特性是故意的, 原因来自于浏览器大战时期, 微软与网景在一些技术实现上故意采用了不同方式, 从而让开发者进入自己的阵营, 打击对手的浏览器。

有一部分是上部分的延续, 当网景失败后, 微软在一些新特性的实现上依然我行我素, 没有参考 W3C 标准, 导致一部分与标准不兼容。

还有一部分是没有标准可依的内容。其实这块比重占得相当大, Ajax 对象也属于这种情况。没有标准可依时, 微软会选择自己已有的技术来实现, 比如 activeX 或者 CSS filter 等, 很多类似这样的特性最终被标准吸纳, 但是采用了另外的实现方式, 导致不兼容。

第 629 道: 请写出兼容所有浏览器清除浮动的代码。

解析:

```
<div class="clearfix">
```

```

        <div class="fl"></div>
<div class="fr"></div>
</div>
.clearfix:after{
visibility:hidden;
display:block;
font-size:0;
content:" ";
clear:both;
height:0;
}
.clearfix { display: inline-table; }
/* Hides from IE-mac */
* html .clearfix { height: 1%; }
.clearfix { display: block; }
/* End hide from IE-mac */

```

第 630 道：写出 IE 和非 IE 浏览器的条件注释写法。

解析：

IE:

```

<!--[if IE]>
<h1> 您正在使用 IE 浏览器 </h1>
<![endif]-->
<!--[if IE 5]>
<h1> 版本 5</h1>
<![endif]-->
<!--[if IE 5.0]>
<h1> 版本 5.0</h1>
<![endif]-->
<!--[if IE 5.5]>
<h1> 版本 5.5</h1>
<![endif]-->
<!--[if IE 6]>
<h1> 版本 6</h1>
<![endif]-->
<!--[if IE 7]>
<h1> 版本 7</h1>
<![endif]-->

```

非 IE:

```

<!--[if !IE]><!-->
<h1> 您不是使用 Internet Explorer</h1>
<!--<![endif]-->

```

最终在非 IE 和特殊的 IE 浏览器下起作用。

也可以使用 `lte lt` 或者 `gt gte` 来判断, 如:

```
<!--[if lte IE 6]>
  在 IE 6 下显示的信息
<![endif]-->

<!--[if IE 6]><!-->
<h1> 您正在使用 Internet Explorer version 6<br />
或者一个非 IE 浏览器 </h1>
<!--<![endif]-->
```

第 631 道: 列举各浏览器之间的一些常见的 CSS、JS 兼容的问题。

解析:

1. HTML 对象获取问题

Firefox: `document.getElementById("idName");`

IE: `document.idname;` 或者 `document.getElementById("idName");`

解决办法: 统一使用 `document.getElementById("idName");`

2. const 问题

【说明】

在 Firefox 下, 可以使用 `const` 关键字或 `var` 关键字来定义常量。

在 IE 下, 只能使用 `var` 关键字来定义常量。

解决方法: 统一使用 `var` 关键字来定义常量。

3. event.x 与 event.y 问题

【说明】

在 IE 下, `even` 对象有 `x,y` 属性。但是没有 `pageX,pageY` 属性;

在 Firefox 下, `even` 对象有 `pageX,pageY` 属性, 但是没有 `x,y` 属性。

解决方法: 使用 `mX(mX = event.x ? Event.x : event.pageX;)` 来代替 IE 下的 `event.x` 或者 Firefox 下的 `event.pageX`。

(4) window.location.href 问题

【说明】

在 IE 或者 Firefox 2.0.x 下, 可以使用 `window.location` 或 `window.location.href;`

在 Firefox 1.5.x 下, 只能使用 `window.location`。

解决方法: 使用 `window.location` 来代替 `window.location.href`。

5. 模态和非模态窗口问题

【说明】

在 IE 下, 可以通过 `showModalDialog` 和 `showModelessDialog` 打开模态和非模态窗口; 而在 Firefox 下则不能。

解决方法: 直接使用 `window.open (pageURL, name, parameters)` 方式打开新

窗口。

如果需要将子窗口中的参数传递回父窗口，可以在子窗口中使用 `window.opener` 来访问父窗口。

例如：

```
var parWin=window.opener;
```

```
parWin.document.getElementById("Aqing").value = "Aqing";
```

第 632 道：Firefox、Chrome 浏览器是否支持 VBScript 脚本？

解析：

我们知道，IE 除了支持 JavaScript 外，还支持 VBScript，而且 JavaScript 可以直接调用 VBScript 中的自定义函数。如果在 VBScript 的自定义函数中再调用 VBScript 的系统函数，这样结合起来就充分发挥了两种语言的优势。

但是，值得注意的是，Firefox、Chrome 是不支持 VBScript 的，也就是说，如果我们的 JavaScript 中混入了 VBScript，我们的程序是不兼容这些浏览器的。

第 633 道：在 IE 的老版本浏览器中，遇到过哪些经典的 Bug，并说出如何解决？

解析：

浮动元素的双倍 margin，在浮动元素中增加一个 `"display:inline"` 属性，这样就可以轻松解决“浮动元素的双倍 margin”的 Bug。

在 Web 页面设计中，为了达到元素的统一渲染的风格，有时我们需要使用 `min-height` 和 `min-width` 来控制元素的最小高度值和最小宽度值。在别的浏览器都运行正常，可唯独这个 IE6 不识别。因此，在使用 `min-height` 和 `min-width` 时，为了达到效果一致，我们要针对 IE6 另作处理。在 IE6 下 `min-width` 解决起来相当简单，但是要顺利解决 `min-height` 就有点麻烦，这里我们主要来看 `min-height` 的解决办法。

一种方法是采用 `"!important"` 来解决，让 `min-height` 在 IE6 下能正常工作，具体代码如下：

```
.demo {
    min-height: 100px;
    height: auto !important; /* 现代浏览器下，内容高度超过 100px 时
    自动获得其高度 */
    height: 100px; /* 此值设置和 min-height 值一样，因为 IE6 下元素
    高度会根据内容本身的高度而定，所以内容高度低于 min-height 值时，为了达
    到 min-height 效果，需要给元素一个显式的高度值 */
}
```

另一种方法是采用子选择器方法来修复。大家都知道 IE6 是不支持子选择器的，所以我们可以使用这个方法来解决，具体代码如下：

```
.demo {
```

```
min-height: 100px;
height: 100px;
}
html>body .demo {
    height: auto; /* 只有现代浏览器才能识别 */
}
```

第 634 道：浏览器标准模式和怪异模式之间的区别是什么？

解析：

所谓标准模式，是指浏览器按 W3C 标准解析执行代码。而怪异模式则是使用浏览器自己的方式解析执行代码，因为不同浏览器解析执行的方式不一样，所以我们称之为怪异模式。浏览器解析时到底使用标准模式还是怪异模式，与网页中的 DTD 声明直接相关，DTD 声明定义了标准文档的类型（标准模式解析），会使浏览器使用相应的方式加载网页并显示，忽略 DTD 声明，将使网页进入怪异模式 (quirks mode)。如果你的网页代码不含有任何声明，那么浏览器就会采用怪异模式解析；如果你的网页代码含有 DTD 声明，浏览器就会按你所声明的标准解析。在标准模式中，IE6 不认识 !important 声明，而 IE7、IE8、Firefox、Chrome 等浏览器认识；在怪异模式中，IE6、IE7、IE8 都不认识 !important 声明，这只是其中一种区别，还有很多其他区别。所以，要想写出跨浏览器的 CSS，你最好采用标准模式。

17.2 “助力 2”：前端优化

第 635 道：Web 脚本开发环境用什么？调试用什么？如果让你来制作一个访问量很高的大型网站，你会如何来管理所有 CSS 文件、JS 与图片？前端优化知识都有哪些？

解析：

关于 Web 脚本开发环境，以前试用过 aptana，现在用 editplus。

在 Firefox 下调试肯定用 Firebug，在 IE 下，需看页面 DOM，可用 IEInspector。管理方法如下：

- 涉及人手、分工、同步。
- 先期团队必须确定好全局样式 (globe.css)、编码模式 (UTF-8) 等；
- 编写习惯必须一致（例如都是采用继承式的写法，单样式都写成一行）；
- 标注样式编写人，各模块都及时标注（标注关键样式调用的地方）；
- 页面进行标注（例如页面模块的开始和结束）；
- CSS 和 HTML 分文件夹并行存放，命名需统一（例如 style.css）；
- JS 分文件夹存放，以该 JS 功能的英文命名；

- 图片采用整合的 images.png, png8 格式文件使用尽量整合在一起，方便将来的管理。

第 636 道：前端优化的知识有哪些？

解析：

(1) 从用户角度而言，优化能够让页面加载得更快，对用户的操作响应更及时，能够给用户提供更友好的体验。

(2) 从服务商角度而言，优化能够减少页面请求数，或者减小请求所占带宽，能够节省可观的资源。

第 637 道：如何优化前端开发？

解析：

前端优化的途径有很多，按力度大致可以分为两类。

第一类是页面级别的优化，例如 HTTP 请求数、脚本的无阻塞加载、内联脚本的位置优化等。

第二类则是代码级别的优化，例如 JavaScript 中的 DOM 操作优化、CSS 选择符优化、图片优化，以及 HTML 结构优化等。另外，本着提高投入产出比的目的，后文提到的各种优化策略大致按照投入产出比从大到小的顺序排列。

第 638 道：列举三种提高网页加载速度的方法和技巧。

解析：

1. 使用良好的结构

可扩展 HTML (XHTML) 具有许多优势，但是其缺点也很明显。XHTML 可能使您的页面更加符合标准，但是由于它大量使用标记（强制性的 `<start>` 和 `<end>`），这意味着浏览器要下载更多代码。所以，事情都有两面性，尝试在您的网页中使用较少的 XHTML 代码，以减小页面大小。如果您确实不得不使用 XHTML，试着尽可能对它进行优化。

2. 不要使布局超载

坚持简约原则：少即是多。页面中充斥着各种类型的图像、视频、广告等，这大大违背实用性原则。

3. 不要使用图像来表示文本

使用图像表示文本的最常见示例就是在导航栏中。美观的按钮更加具有吸引力，但是它们的加载速度很慢。此外，图像不能由搜索引擎直接索引，因此，使用图像进行导航不利于搜索引擎优化（Search Engine Optimization, SEO）。当使用大量 CSS 技巧就能创建漂亮的按钮时，绝不使用图像来表示文本。

第 639 道：请写出三种减少页面加载时间的方法？

解析：

1. CSS 格式定义放置在文件头部

这项设置对于用户端是慢速网络或网页内容比较庞大的情况比较有利，它

在网页逐步呈现的同时仍会保持格式信息，不影响网页美感。

2. JavaScript 脚本放在文件末尾

很多 JavaScript 脚本执行效率低下，或者有的第三方域名脚本出现意外无法载入，如果将这些脚本放置到页面比较靠前的位置，可能会导致网站的内容载入速度下降，甚至无法正常加载，所以一般将这些脚本放置在网页文件末尾。一定要放置在前面的脚本，要改用所谓的“后载入”方式加载，即在主体网页加载完成后再加载，防止其影响到主体网页的加载速度。

3. Ajax 调用尽量采用 GET 方法

实际使用 XMLHttpRequest 时，如果使用 POST 方法实现，会发生两次 HTTP 请求。如果改用 GET 方法，只会发生一次 HTTP 请求，该请求减少 50%！

第 640 道：简单地说一下你会从哪些方面优化一个页面？

解析：

- (1) 网站布局优化：网站导航、页面布局、文件目录、地址栏等。
- (2) 网页标签优化：网页 title 关键字标签、网页简介标签、图像注释等。
- (3) 网页代码优化：对网页代码瘦身压缩，调整网页代码的布局，增强代码的可读性和健壮性，调整页面的大小，加快页面阅读速度。
- (4) 超链接优化：超链接布局、超链接注释、超链接途径优化，使之便于搜索引擎收录和用户的阅读。
- (5) 页面内容优化：对主要页面内容进行调整，对排版进行优化，让内容更容易阅读。
- (6) 关键字优化部署优化关键字，调整关键字呈现的顺序和呈现的位置，使之契合搜索引擎蜘蛛的抓取习惯，增强搜索引擎的友好性，适当加大关键字的密度，使网站排名靠前。

第 641 道：针对大量图片的展现，请说出其性能优化的点都有哪些？

解析：

针对这个问题，主要从以下两个方面来进行描述：

从图片的处理方面来说，图片一定要清晰，能够准确地显示出产品的特点和客户关心的问题。在保证图片质量的前提下，要尽量的小，多跟美工商量，把图片文件的容量尽可能地改小，这样不仅对搜索引擎比较好，同时也缩短了页面加载的时间，不至于让那些没有耐心等待的顾客白白跑掉。

从图片 alt 方面来说，搜索引擎很难识别图片内容，所以只有告诉它这个图片是什么，这就是 alt 的职责所在了，同时，如果这张图片打不开或已失效，它又能告诉访客这大概是什么。如果访客对该产品感兴趣，自然会通过网站上留下的联系方式与相关人员取得联系。如果一个页面有多张图片，不要写相同的 alt，可以融入相关的长尾关键词。

第 642 道：网速性能优化的方法有哪些？

解析：

对于优化网速性能，有以下几种方法：

- (1) 检查网络速度。
- (2) 查看电脑中是否安装了一些 P2P 技术软件。
- (3) 优化电脑系统。

第 643 道：如何对网站的文件和资源进行优化？

解析：

- (1) 文件合并。
- (2) 文件最小化 / 文件压缩。
- (3) 使用 CDN 托管。
- (4) 缓存的使用。

第 644 道：简述交互设计与传统设计的区别。交互设计的核心是什么？

解析：

交互设计与传统设计的区别：

- (1) 使用场景的复杂。
- (2) 使用时间碎片化。
- (3) 屏幕尺寸的缩小。
- (4) 无法多任务地处理信息。
- (5) 平台的设计规范和特性。

交互设计的核心：

(1) 交互是两个实体之间的事务，通常涉及信息的交换，也可包括实物或服务的交换。

(2) 交互发生在人、机器和系统之间，存在多种不同的组合。

17.3 “助力 3”：开发者工具

第 645 道：在页面设计中，你通常用到哪些软件？

解析：

Macromedia Dreamweaver 8 是建立 Web 站点和应用程序的专业工具。它可将可视布局工具、应用程序开发功能和代码编辑支持组合在一起，其功能强大，使得各个层次的开发人员和设计人员都能够快速创建标准的、界面很吸引人的网站和应用程序。从基于 CSS 的设计的领先支持到手工编码功能，Dreamweaver 提供了专业人员在集成、高效的环境中所需的工具。开发人员可以使用 Dreamweaver 及所选择的服务器技术来创建功能强大的 Internet 应用程序，从而

使用户能连接到数据库、Web 服务和旧式系统。

第 646 道：经常使用的页面开发工具和测试工具有哪些？

解析：

(1) Loderruner 测试工具是测试性能问题的，对于页面流测试不能展现，不直观，不能使用。

(2) QuickTest Professional 页面操作复杂，使用测试时失败率很高，对于平台的一些事件无法获取，一旦有弹出页面，整个测试就会报错，对于现在的需求页面流来说，实施成本太大。

(3) 对于 Tellurium 与 Selenium，使用过程中发现功能很齐全，基本可以实现其他页面流测试的所有功能，并且还能结合 jQuery 等 JS 框架操作控件，但是在抓脚本时需要用火狐浏览器的插件，这对于平台应用来说限制较多，此工具可以通过录制脚本获取页面上的控件及控件的各种事件。Tellurium 比 Selenium 更进步，只要页面上控件的 id 位置不变，修改脚本就比较简单，因为它不依赖控件的类型，Tellurium 对脚本维护量不大，不过前期录制脚本的成本太高，实现起来并不现实，所以暂不推荐使用此工具，此工具对于研发人员来说，进行自己的页面流测试比较适合。

(4) OracleATS，全称 Oracle Application Testing Suite。

现在 OracleATS 在性能测试中支持录制 / 回放的只有：

- HTTP 的脚本录制和压力测试；
- 基于 HTTP 协议的 Web Service 的脚本录制和压力测试；
- 基于 EBS Forms 的脚本录制和压力测试。

第 647 道：你更喜欢在哪个浏览器下进行开发？你使用哪些开发工具？

解析：

这类面试题比较主观，应从自身角度来回答。

常用的浏览器有：Firefox、Chrome、IE、Opera、Safari、遨游、360。

常用的开发工具有：Firebug、HttpWatch、Fiddler、HttpFox、Yslow。

第 648 道：你常用的开发环境是怎样的？顺便说说操作系统、文本编辑器、浏览器，以及其他工具等方面。

解析：

Windows 7；

closure-compiler：JS 压缩工具（Google 出品）；

yuicompressor：yui 压缩工具，不仅能压缩 JS，还能压缩 CSS；

pngout：png 图片压缩工具；

cssEmbed：将图片转为 base64 嵌入代码；

YSlow：Firefox 插件，它能把整个页面中所有的 JavaScript、CSS 等合并并

一个文件；

measureIt: Firefox 插件，起着一把尺子的作用，它能精确看到部分元素之间的距离；

idebug: 淘宝内部的一个测试工具。

第 649 道：常用的前端开发工具有哪些？

解析：

常用的开发工具有：

- (1) Firebug。
- (2) HttpWatch。
- (3) Fiddler。
- (4) HttpFox。
- (5) Yslow。
- (6) CSS Usage。
- (7) VIM。
- (8) Editplus。
- (9) DNS Flusher。
- (10) PageSpeed。

17.4 “助力 4”：JS 库和框架

第 650 道：你是否研究过你所使用的 JS 库或者框架的代码？

解析：

framework 主要是给用户使用的，当然，也给二次开发者使用。它易使用、易扩展、可持续发展、可控。

有些框架，依赖它开发的东西，只能在它的“树荫”下生存，而无法独立。这样的框架，就不能当库来用，如 jQuery。

有些库，几乎没有框架的特性，所以易用性很差，如 YUI2。所以，它的用户群也只能局限于二次开发者，或水平比较高的开发者。

作为框架，可能为了易用性，牺牲了严谨性。例如：渲染 String 与 Array 的 prototype。这就与库的无污染理念冲突。

所以，一个产品，如果它自己既能当框架，也能当库，那么，它就需要权衡取舍。当然，它也可以选择有多种形式的输出（输出成独立的库，或是易用的框架）。

第 651 道：常用 JS 框架有哪些？jQuery 的优点有哪些？

解析：

常用的 JS 框架有：Dojo、Scriptaculous、Prototype、yui-ext、jQuery、Mochikit、mootools、moo.fx。

jQuery 的优点有以下 10 个：

- (1) 轻量级。
- (2) 强大的选择器。
- (3) 出色的 DOM 操作的封装。
- (4) 可靠的事件处理机制。
- (5) 完善的 Ajax。
- (6) 不污染顶级变量。
- (7) 出色的浏览器兼容性。
- (8) 链式操作方式。
- (9) 行为层与结构层的分离。
- (10) 丰富的插件支持。

第 652 道：简单介绍你所用过或了解的前端框架，以及它们的特点。

解析：

在这里主要介绍一下 Bootstrap 框架。

由匠人造，为匠人用。

与所有前端开发人员一样，Bootstrap 团队是由国际上最优秀的前端开发组织的，他们乐于创造出色的 Web 应用，同时希望帮助更多同行从业者，为同行提供更高效、更简洁的产品。

1. 适应各种技术水平

Bootstrap 适应不同技术水平的从业者，无论是设计师，还是程序开发人员；不管是骨灰级别的大牛，还是刚入门的菜鸟。使用 Bootstrap 既能开发简单的小东西，也能构造更为复杂的应用。

2. 跨设备、跨浏览器

最初设想的 Bootstrap 只支持现代浏览器，不过新版本已经能支持所有主流浏览器，甚至包括 IE7。从 Bootstrap 2 开始，提供对平板和智能手机的支持。

3. 提供 12 列栅格布局

栅格系统不是万能的，不过在应用的核心层有一个稳定和灵活的栅格系统，确实可以让开发变得更简单。可以选用内置的栅格，或是自己手写。

4. 支持响应式设计

从 Bootstrap 2 开始，提供完整的响应式特性。所有的组件都能根据分辨率和设备灵活缩放，从而提供一致性的用户体验。

5. 样式化的文档

与其他前端开发工具包不同，Bootstrap 优先设计了一个样式化的使用指南，

不仅可以用来介绍特性，而且可以用来展示最佳实践、应用，以及代码示例。

6. 不断完善的代码库

尽管经过 gzip 压缩后，Bootstrap 只有 10KB 大小，但是它却仍是最完备的前端工具包之一，提供了几十个全功能的随时可用的组件。

7. 可定制的 jQuery 插件

任何出色的组件设计，都应该提供易用、易扩展的人机界面。Bootstrap 为此提供了定制的 jQuery 内置插件。

8. 选用 LESS 构建动态样式

当传统的枯燥 CSS 写法止步不前时，LESS 技术横空出世。LESS 使用变量、嵌套、操作、混合编码，帮助用户花费很少的时间成本，编写更快、更灵活的 CSS。

9. 支持 HTML5

Bootstrap 支持 HTML5 标签和语法，要求建立在 HTML5 文档类型基础上进行设计和开发。

10. 支持 CSS3

Bootstrap 支持 CSS3 的所有属性和标准，逐步改进组件以达到最终效果。

11. 提供开源代码

Bootstrap 全部托管于 GitHub (<https://github.com/>)，完全开放源代码，并借助 GitHub 平台实现社区化开发和共建。

12. 由 Twitter 制造

Twitter 是互联网的技术先驱，引领时代技术潮流，Twitter 前端开发团队是公认的最棒的团队之一，整个 Bootstrap 项目由经验丰富的工程师和设计师奉献。

第 653 道：你最熟悉的 Web 的前端框架、库有哪些？

解析：

1. jQuery

jQuery 是对 JS 底层 DOM 操作封装最薄的一个框架，没有大量的专有对象，多为提供函数进行 DOM 操作。准确地说，它不是偏重于客户端的框架，而是侧重于对 jsdom 编程。

2. ExtJS

ExtJS 可以用来开发 RIA，它也是客户端的 Ajax 应用，是用 JavaScript 写的，主要用于创建前端用户界面，是一个与后台技术无关的前端 Ajax 框架。

3. Flex

Flex 是一个高效、免费的开源框架，它与 Ext 不同，它有健壮的可视化开发工具 Flash Builder，可以同 C# 一样进行拖曳布局，生成一种 XML，也便于维护，编译后生成 swf 文件，直接嵌入 HTML 即可。为提高安全性，浏览时同

Flash 一样，需要 Flash Player。

4. Silverlight

微软 Silverlight 是一个跨浏览器、跨客户平台的技术，能够设计、开发和发布有多媒体体验与富交互的网络交互程序。主要是应用在微软系列的语言中，包括 CS 与 BS 架构。

第 654 道：说一下自己最熟悉的 JS 框架的结构及原理。

解析：

1. jQuery

查找（创建）jQuery 对象：`$("selector")`；

调用 jQuery 对象，完成我们的工作：`$("selector").doOurWork()`；

jQuery 就是以这种最简单的编码逻辑来改变 JavaScript 编码方式的。这两个步骤是 jQuery 的编码逻辑核心。

要实现这种简洁编码方式，创建 jQuery 对象这一环节至关重要。因此，jQuery 的 DOM 元素查找能力相当强大。此外，jQuery 对象的方法肯定是有限的，有限的方法满足不了日益增长的各种要求，所以，必须提供 jQuery 对象方法的扩展能力。

强大的 DOM 元素查找能力，以及随心所欲的方法的扩展，这两点正是 jQuery 的核心所在！

2. ExtJS

ExtJS 是一个开源软件，从性质来讲，和 jQuery 差不多，都是基于 JS 的方便开发的封装类库。

它的优势是将一些图形界面效果封装到代码中，作为类存在，使用时直接 new 或者 create 就可以得到一个不错的组件。类似于 Java 中的 swing。

它对于组件的定义比较清晰，方便进行事件绑定和 DOM 操作，简化 JS 逻辑，例如封装分页、Ajax 请求。

Ext 绝对可以单独使用。实际上，除了有特定的要求外，我们推荐单独使用 Ext，这样的话文件占位更小，支持和整合也更紧密。我们也支持与 jQuery、YUI 或 Prototype 整合使用，作为低层库的角色出现，以提供处理各种核心的服务，如 DOM 和事件处理，Ajax 连接和动画特效。使用整合方式的一个原因是，它们已具备了一些特定的器件，而 Ext 并没有原生支持——像 YUI 的 History 控件便是一个典型的例子。这时，Ext 需要依赖 YUI 这个库的底层来实现 History 控件，这样也可免去 Ext 自身底层库，从而减少整个程序的内存占用。另一个使用整合方式的原因是，对于许多已在使用其他底层库的程序，可能希望逐步加入 Ext。总之，如果已经有了其他库，Ext 可已利用它们。我们的宗旨是为用户提供各种可能性和性能上的优化。而事实是，只要实现了相对应的底层库接口，为任意一个框架添加上适配器是没有问题的——人们可以轻松地将 Dojo、

Moo，或其他 JS 库转变为 Ext 的底层。

控件其实也是组件，比如用于显示树信息的 TreePanel、用于显示表格的 GridPanel 及 EditorGridPanel，还有代表应用程序窗口的 Ext.window 等都属于 Ext 控件。在使用 Ext 时，一定要掌握一些核心控件，特别是处于基类的控件。比如上面提到的几个控件，它们都是继承于面板 Panel，所以我们要重点掌握面板这个核心控件的特性。比如面板由一个顶部工具栏（tbar）、一个底部工具栏（bbar）、面板头部（header）、面板尾部（bottom）、面板主区域（body）几个部分组成。面板类中还内置了面板展开、关闭等功能，并提供一系列可重用的工具按钮，使我们可以轻松实现自定义的行为，面板可以放入其他任何容器中，面板本身是一个容器，它里面又可以包含各种其他组件。只要掌握了 Panel 的应用，那么学习 TreePanel、window 等就会简单得多。

同样的道理，对于 Ext 的表单字段来说，不管是 ComboBox、NumberField，还是 DateField，它们都是 Ext、Form、Field 类的子类，上面定义了表单字段的各种基本操作及特性。在学习使用表单字段组件时，一定要重点研究 Field 这个类，掌握它的主要方法、事件等，这样有助于更好地学习使用其他的字段。

第 655 道：写出几个除 jQuery 以外的 JS 框架名称。

解析：

```
Yui-ext
Dojo
Prototype
GooJS
Zepto
DOMBuilder
```

第 656 道：你是否研究过你所使用的 JS 库或框架的源代码？

解析：

在这里主要介绍一下 ExtJS 框架。

(1) Ext 获取一个 HTML 文档里的节点

```
Ext.onReady(function(){
    Var mydiv=Ext.get(“mydiv”));
```

(2) 在页面里 ID 为 myButton 的按钮加上了一个 click 事件，触发这个事件以后会弹出一个提示框。

```
Ext.onReady(function(){
    Ext.get('myButton').on('click',function(){
        alert("button");
    });
});
```

第 657 道：简述一下你对 HTML 语义化的理解？

解析：

HTML 语义化是指根据内容的结构化（内容语义化），选择合适的标签（代码语义化），便于开发者阅读和写出更优雅的代码，同时让浏览器的爬虫和机器很好地解析代码。

HTML 语义化的主要目的是：

- 为了在没有 CSS 的情况下，页面也能呈现出很好的内容结构和代码结构；
- 有利于用户体验；
- 有利于 SEO 和搜索引擎建立良好的沟通；
- 方便其他设备，以有意义的方式来渲染网页；
- 便于团队开发和维护，增加可读性。

第 658 道：PhoneGap 的作用是什么？

解析：

PhoneGap 是一个用于创建移动跨平台应用程序的快速开发平台。只要你会 HTML 和 JavaScript，或者 Java 语言，就可以利用 PhoneGap 提供的 API 去调用各种功能，PhoneGap 可以让你制作出在各种手机平台上运行的应用。

第 659 道：Backbone 可单独运行吗？是否要依赖库？如果依赖，那么它依赖哪个？

解析：

Backbone.JS 提供了一套 Web 开发的框架，通过 Models 进行 key-value 绑定及 custom 事件处理，通过 Collections 提供一套丰富的 API 实现枚举功能，通过 Views 来进行事件处理，以及与现有的 Application 通过 RESTful JSON 接口进行交互。

Backbone.JS 是基于 jQuery 和 underscore 的一个 JS 框架。所以，它依赖于 jQuery。

第 660 道：SeaJS 的作用是什么？

解析：

使用 SeaJS，可以规范模块的书写格式，并能自动处理模块的依赖，还非常有助于代码组织、开发调试和性能优化。SeaJS 期待能给你提供简单、极致的模块化开发体验。解决网站开发中的问题，如冲突、性能、依赖等。

第 661 道：Zepto 的作用是什么？其默认包含哪些模块？是否支持所有的主流浏览器？

解析：

Zepto 只针对移动端浏览器编写，因此其体积更小、效率更高。更重要的是，它的 API 完全仿照 jQuery，所以学习成本也很低。

Zepto 默认包含 touch 模块。

郑重提醒：text :checkbox :first 等在 jQuery 里是很常用的选择器，但 Zepto 不支持。原因很简单，jQuery 通过自己编写的 sizzle 引擎来支持 CSS 选择器，

而 Zepto 直接通过浏览器提供的 `document.querySelectorAll` 接口来支持 CSS 选择器，且这个接口只支持标准的 CSS 选择器，而上面提到的那些属于 jQuery 选择器扩展，所以仔细看看网页，注意一下这些选择器。当然也有好消息，如果有 `selector` 这个模块的话，那么就能够支持部分的 jQuery 选择器扩展，列举如下：

```
:visible :hidden
:selected :checked
:parent
:first :last :eq
:contains :has
```

第 662 道：写出 PhoneGap 和 AppCan 的区别？

解析：

PhoneGap：

PhoneGap 是一个基于 HTML、CSS 和 JavaScript 的创建跨平台移动应用程序的快速开发平台。它使开发者能够利用 iPhone、Android、Palm、Symbian、WP7、Bada 和 Blackberry 智能手机的核心功能——包括地理定位、加速器、联系人、声音和振动等。此外，PhoneGap 拥有丰富的插件，可以以此扩展无限的功能。PhoneGap 是免费的，但是它需要特定平台提供的附加软件，例如 iPhone 的 iPhone SDK、Android 的 Android SDK 等，也可以和 DW5.5 配套开发。使用 PhoneGap 比为每个平台分别建立应用程序好一点，因为虽然基本代码是一样的，但是却仍然需要为每个平台分别编译应用程序。

AppCan：

AppCan 是国内 Hybrid App 混合模式开发的倡导者，AppCan 应用引擎支持 Hybrid App 的开发和运行，并且着重解决了基于 HTML5 的移动应用中目前“不流畅”和“体验差”的问题。使用 AppCan 应用引擎提供的 Native 交互功能，可以让 HTML5 开发的移动应用基本接近 Native App 的体验。

第 663 道：简述 Angular 的双向数据绑定。

解析：

这里有一个例子供参考。

Angular：

```
<div ng-controller="CounterCtrl">
  <span ng-bind="counter"></span>
  <button ng-click="counter++">increase</button>
</div>
function CounterCtrl($scope) {
  $scope.counter = 1;
}
```

17.5 “助力 5”：cookie

第 664 道：cookie 是什么？在使用 cookie 时有哪些需要注意的地方？

解析：

1. cookie 是临时文件的意思，它会保存你浏览网页的痕迹，使你再次上同一页面时能提高网速，它会判断你是否登录过此网站，有些是可以帮你自动登录的。

cookie 有时也用其复数形式 Cookies，指某些网站为了辨别用户身份，进行 session 跟踪而储存在用户本地终端上的数据（通常经过加密）。

2. cookie 注意事项

(1) SetCookie() 之前不能有任何 HTML 输出，它认了第二，没有哪个元素敢认第一，就是空格和空白行也不行。

(2) SetCookie() 执行之后，在当前页调用 `echo $_Cookie["name"]` 不会有输出。必须刷新或到下一个页面才可以看到 cookie 值。原因很简单：SetCookie() 执行之后，往客户端发送一个 cookie，你不刷新或浏览下一个页面，客户端怎么把 cookie 给你送回去，浏览器创建了一个 cookie 后，对于每一个针对该网站的请求，都会在 Header 中带着这个 cookie。不过，对于其他网站的请求，cookie 是绝对不会跟着发送的，而且浏览器会一直这样发送，直到 cookie 过期为止。

(3) 使用 cookie 的限制。一个浏览器能创建的 cookie 数量最多为 30 个，并且每个不能超过 4KB，每个 Web 站点能设置的 cookie 总数不能超过 20 个。

(4) cookie 是保存在客户端的，如果用户禁用了 cookie，那么 cookie 自然也就没作用了。现在的浏览器，每当发送一个 cookie 给客户端，就会被拦截住，并询问用户是否允许 cookie 进入。

第 665 道：在 Web 开发中，我们会经常用到 cookie，那么如何设置一个 cookie 呢？对于性能的影响又是怎样的？

解析：

浏览器负责管理用户系统上的 cookie。Cookie 通过 `HttpResponse` 对象发送给浏览器，该对象被公开称为 cookies 的集合。可以将 `HttpResponse` 对象作为 `Page` 类的 `Response` 属性来访问。要发送给浏览器的所有 cookie 都必须添加到此集合中。创建 cookie 时，需要指定 Name 和 Value。每个 cookie 必须有一个唯一的名称，以便从浏览器读取 cookie 时可以识别它。由于 cookie 按名称存储，因此用相同的名称命名两个 cookie 会导致其中一个 cookie 被覆盖。

```
Response.Cookies["userName"].Value = "patrick";
Response.Cookies["userName"].Expires = DateTime.Now.
AddDays(1);
```

直接设置 Cookies 集合的值，可以通过这种方式向集合添加值，因为 Cookies 是从 NameObjectCollectionBase 类型的专用集合派生的。

```
HttpCookie aCookie = new HttpCookie("lastVisit");
aCookie.Value = DateTime.Now.ToString();
aCookie.Expires = DateTime.Now.AddDays(1);
Response.Cookies.Add(aCookie);
```

创建了一个 HttpCookie 类型的对象实例，设置其属性，然后通过 Add 方法将其添加到 cookies 集合。在实例化 HttpCookie 对象时，必须将该 Cookie 的名称作为构造函数的一部分进行传递。

17.6 “助力 6”：超级素数

第 666 道：判断一个数是否为超级素数。

解析：

```
int sushu(int N){
    int i;
    int flag=1;
    if (N==1) return false;//1 既不是素数也不是质数
    if (N==2) return true;
    for (i=2;i<=sqrt(N);i++){
        if (N%i==0) {
            flag=0
            break;
        }
    }
    return flag;
}
```

17.7 “助力 7”：主流技术

第 667 道：请简述建设网站涉及的主要方面，当前开发使用的主流技术及趋势。

解析：

(1) 建设网站涉及的主要方面有：

- 网站关键词的优化；
- 网站结构优化；
- 网站页面优化；

- 网站内容更新;
- 网站外部链接建设;
- 网站流量分析。

(2) 大体来说, 分为三个流派, 即 Java Web、.NET、PHP。

具体的技术有以下几个。

前端的: JS、HTML、CSS、PS、Flash……

一些主流的技术: Ajax、jQuery。

后台的 Java (PHP、.NET)、SQL、服务器。

如 Java 里面: JSP、Servlet。

值得一提的还是 Java 的三大开源框架: Struts2、Spring、Hibernate。

(3) 随着计算机应用的日益普遍和系统规模的扩大, 前端开发得到了长足的发展。开发服务技术及其发展趋势有了很大的提高。

Web 开发技术是目前网络应用开发技术的最重要的成员之一。随着服务器端应用的普及, 客户端应用大军开始向微软的 .NET 战略行进, 以迎接 Web 开发技术新革命的到来。

17.8 “助力 8”: 进制转化

第 668 道: 把十六进制颜色转变成 RGB 颜色, 如 :#FFFFFF 等同 RGB (255,255,255)

解析:

```
function exChange(color)
{
    return 'rgb(' + parseInt(color.substr(1, 2), 16) + ',' +
           parseInt(color.substr(3, 2), 16) +
           ',' +
           parseInt(color.substr(5, 2), 16)
           + ')';
}
```

第 669 道: 写一段 JS 程序, 实现将十进制数 302 转换为二进制数。

解析:

```
string ByteToBinaryString(char v) {
    if (v == 1) {
        return "1";
    }
    if (v == 0) {
        return "0";
    }
    if (v % 2 == 0) {
        return ByteToBinaryString(v / 2) + "0";
    }
}
```

```

    } else {
        return ByteToBinaryString(v / 2) + "1";
    }
}

```

17.9 “助力 9”：追加字符串

第 670 道：在 DOM 中获取一个 `<div id='mydiv'>` 元素，并且在该元素原有内容基础上，结尾追加字符串“END”。（分别使用标准 JavaScript 语言及 jQuery 实现以上功能）

解析：

```

var oMydiv = document.getElementById( " mydiv" );
var oMydivText = oMydiv.innerHTML;
oMydiv.innerHTML = oMydivText + "END";
var oMydiv = $( "#mydiv" ).html();
$( "#mydiv" ).html(oMydiv + "END");

```

17.10 “助力 10”：模块模式

第 671 道：请解释什么是 JavaScript 的模块模式，并举出实例。

解析：

模块模式主要用来为单例创建私有变量和特权方法（公有方法），从而增强单例的可访问性。以模块模式定义的私有变量和私有函数，只能使用单例对象本身的特权（公有）方法才可以访问到，其他外部的任何对象都不可以。

```

var singleton = function() {
    // 私有变量
    var privateVariable = 10;
    // 私有函数
    function privateFunction() {
        return false;
    }
    // 返回对象
    return {
        // 公有属性
        publicProperty: true,
        // 公有属性和公有方法
        publicMethod : function() {
            privateVariable++;
        }
    };
};

```

```
    return privateFunction();  
  }  
};  
}();
```

第 672 道：写出你对 CMD/AMD 模块化的理解。

解析：

CMD，全称即 Common Module Definition。

在 CMD 规范里，一个模块就是一个 JavaScript 文件。

CMD 与 AMD 的区别：

在执行顺序上，CMD 是延迟执行的。而 AMD 是提前执行的。requireJS 从 2.0 开始可以延迟执行。

第 673 道：AMD 模式是什么？

解析：

AMD（Asynchronous Module Definition，异步模块定义）是用于异步加载的一种模块定义方式。随着 Web 应用不断发展和对 JavaScript 依赖的进一步加深，出现了使用模块（Modules）来组织代码。模块使我们能够创建明确清晰的组件和接口，这些组件和接口很容易加载并连接到其依赖组件。AMD 模块系统提供了使用 JavaScript 模块来构建 Web 应用的完美方式，并且这种方式具有形式简单、异步加载，并且被广泛采用的特点。AMD 格式是一套 API，它用于定义可重用的并能在多种框架使用的模块。开发 AMD 是为了提供一种定义模块的方式，这种方式可以使用原生的浏览器脚本元素机制来实现模块的异步加载。AMD API 是由 2009 年 Dojo 社区的讨论中产生的，然后移动到讨论 CommonJS 如何更好地为浏览器适应 CommonJS 模块格式（被 NodeJS 使用）。CommonJS 已经发展成为单独的一个标准并有其专门的社区。AMD 已经广泛普及，形成了众多模块加载实现并被广泛使用。AMD 格式总体的目标是为现在的开发者提供一个可用的模块化。

AMD 格式本身是一个关于如何定义模块的提案，在这种定义下，模块和依赖项都能够异步地进行加载。它有很多独特的优势，包括天生的异步及高度灵活等特性，这些特性能够解除常见的代码与模块标识间的那种紧密耦合。目前它已经被很多项目所接纳，包括 jQuery。

第 674 道：简述模版引擎的作用。

解析：

使我们的数据和表现层分离，便于理解和后期的维护。

第 675 道：请解释一下 CommonJS 与 AMD ？

解析：

CommonJS 是 JavaScript 模块化编程的一种规范，并且主要是在服务器端模

块化的规范，一个单独的文件就是一个模块。每一个模块都是一个单独的作用域，也就是说，在该模块内部定义的变量，无法被其他模块读取，除非定义为 global 对象的属性。CommonJS 加载模块是同步的，所以只有加载完成以后，才能执行后面的操作。Node.js 采用了这个规范，Node.js 主要是用于服务器的编程加载的模块文件，一般都已经存在本地硬盘，所以加载起来比较快，不用考虑异步加载的方式，所以 CommonJS 规范比较适用。但如果根据浏览器环境，要从服务器加载模块，这时就必须采用异步模式。所以就有了 AMD CMD 解决方案。

在网站性能优化正在逐步成为产业的今天，CommonJS 的 Module/Transport 规范里，目前认可度最高的是 Modules/AsynchronousDefinition（简称 AMD），意思就是“异步模块定义”。异步方式加载模块，模块的加载不影响它后面语句的执行。所有依赖这个模块的语句，都定义到一个回调函数中，等加载完成之后，这个回调函数才会运行。

17.11 “助力 11”：效果题

第 676 道：如何用 JS 实现一个选项卡效果，可以用任何你会的 JS 框架，如 jQuery 等。

解析：

```
<script type="text/javascript">
    window.onload = function () {
        var oLi = document.getElementById("tab").
        getElementsByTagName("li");
        var oUl = document.getElementById("content").
        getElementsByTagName("ul ");
        for(var i = 0; i < oLi.length; i++)
        {
            oLi[i].index = i;
            oLi[i].onmouseover = function () {
                for(var n = 0;n < oLi.length;n++)
                {
                    oLi[n].className = "";
                    this.className = "current";
                }
            }
            for(var n = 0;n <oUl.length;n++)
            {
                oUl[n].style.display = "none";
```

```

        oUl[this.index].style.display="block";
    }
}
}
}
</script>

```

第 677 道：编写一段代码，实现类似百度搜索输入框智能提示功能，包括程序逻辑实现，以及前端实现。

解析：

```

$(function(){
    // 取得 div 层
    var $search = $('#search');
    // 取得输入框 jQuery 对象
    var $searchInput = $search.find('#search-text');
    // 关闭浏览器提供给输入框的自动完成
    $searchInput.attr('autocomplete','off');
    // 创建自动完成的下拉列表，用于显示服务器返回的数据，插入在搜索按钮
    的后面，等显示的时候再调整位置
    var $autocomplete = $('
')
        .hide()
        .insertAfter('#submit');
    // 清空下拉列表的内容，并且隐藏下拉列表区
    var clear = function(){
        $autocomplete.empty().hide();
    };
    // 注册事件，当输入框失去焦点时清空下拉列表并隐藏
    $searchInput.blur(function(){
        setTimeout(clear,500);
    });
});

```

第 678 道：用 JS、HTML、CSS 实现一个弹出提示控件（原生）。

解析：

(1) 分别实现类似于系统的 alert、confirm、prompt 对话框。

```

script type="text/JavaScript">
    alert(' 这是 alert 提示框 ');
    confirm(' 这是确定框 ');
    prompt(' 请输入内容 ', ' 输入框 ');
</script>

```

(2) 对话框的大小根据提示内容进行自适应（有一个最小宽高），默认为在页面的水平垂直居中。

```

<style type=text/css>

```



```
.block {
  text-align: center;
  background: #c0c0c0;
  border: #a0a0a0 solid 1px;
  margin: 20px;
}
.block:before {
  content: '';
  display: inline-block;
  height: 100%;
  vertical-align: middle;
  margin-right: -0.25em;
}
.centered {
  display: inline-block;
  vertical-align: middle;
  width: 300px;
  padding: 10px 15px;
  border: #a0a0a0 solid 1px;
  background: #f5f5f5;
}
```

(3) 对话框可拖动。

```
var objMove = null;
var pX = 0;
var pY = 0;
document.onmouseup = MUp;
document.onmousemove = MMove;
function MDown(objMoveId)
{
    var hitpoint = event.srcElement;
    if( hitpoint.tagName == "INPUT"
    || hitpoint.tagName == "TEXTAREA"
    || hitpoint.tagName == "SELECT" )
    {
        objMove = null;
        return;
    }

    objMove = document.getElementById(objMoveId);
    if( objMove == null )
    {
        return;
    }
}
```

```

    }

    objMove.style.cursor = "move";
    //objMove.setCapture();
    pX = event.x - objMove.style.pixelLeft;
    pY = event.y - objMove.style.pixelTop;
}

function MMove()
{
    if(objMove != null)
    {
        objMove.style.left = event.x - pX;
        objMove.style.top = event.y - pY;
    }
}

function MUp()
{
    if(objMove != null)
    {
        //objMove.releaseCapture();
        objMove.style.cursor = "default";
        objMove = null;
    }
}

```

(4) 对话框中的事件，可以模拟系统对话框的事件（例如：alert 对话框，当单击确定按钮时，对话框消失）。

采用 ClassWizard 实现重载 OnOK()。

```

OnOK(){ if(...)
        { // 当按下 Enter 或 Esc 按键事件时 } else
        { // 单击关闭按钮时
            CDialog::OnOK();
        }
}

```

(5) 解决 IE6 被 select 控件遮挡的问题。

frame 是 IE5.5 以后提出来的一个 window 组件，在 iframe 中只能嵌套页面，所以 iframe 中有 src 属性可以直接引用一个页面。在 IE 中有一组规范，iframe 的优先级要比 select 高，所以在页面显示时，如果 iframe 和 select 在同一个位置的话，我们会发现，iframe 把 select 遮挡了，而 div 中可以加入 iframe，有了这个基础，我们可以在显示的 div 层中加入一个空的 iframe。

```
<iframe style="position: absolute; z-index: -1; width: 100%; height: 100%; top: 0; left: 0; scrolling: no;" frameborder="0" src="about:blank">
```

17.12 “助力 12”：跨域问题

第 679 道：关于数据的跨域问题，请提出几个通过 JavaScript 实现的解决方案并附上简单代码。

解析：

对于主域相同而子域不同的问题，可以通过设置 `document.domain` 的办法来解决。具体的做法是，可以在 `http://www.a.com/a.html` 和 `http://script.a.com/b.html` 两个文件中分别加上 `document.domain = 'a.com'`；然后通过 `a.html` 文件中创建一个 `iframe`，去控制 `iframe` 的 `contentDocument`，这样两个 JS 文件之间就可以“交互”了。当然这种办法只能解决主域相同而二级域名不同的情况，如果你把 `script.a.com` 的 `domain` 设为 `alibaba.com`，那显然是会报错的。

这种方式适用于 `{www.qdihu.com, qdihu.com, js.qdihu.com, css.qdihu.com}` 中的任何页面相互通信。

备注：某一页面的 `domain` 默认等于 `window.location.hostname`。主域名是不带 `www` 的域名，例如 `a.com`，主域名前面带前缀的通常都为二级域名或多级域名，例如 `www.a.com` 其实是二级域名。`domain` 只能设置为主域名，不可以在 `b.a.com` 中将 `domain` 设置为 `c.a.com`。

17.13 “助力 13”：前端交谈

第 680 道：如何提高前端性能，至少写出三点。

解析：

1. 尽量减少 HTTP 请求

前端优化的黄金准则指导着前端页面的优化策略：只有 10% ~ 20% 的最终用户响应时间花在接受请求的 HTML 文档上，剩下的 80% ~ 90% 时间花在为 HTML 文档所引用的所有组件（图片、脚本、样式表等）进行的 HTTP 请求上。因此，改善响应时间最简单的途径就是减少组件的数量，并由此减少 HTTP 请求的数量。所以，我们要尽量减少 HTTP 请求。

2. 压缩组件（使用 Gzip 方式）

对于做 PC 网站或者移动网站来说，急需要压缩的是 CSS 文件和 JS 文件，至于如何压缩，网上有很多在线工具，去挑选一个自己用得顺手看得顺眼的就

好，当然，也有人选择对 HTML 进行压缩，这样也可以。

3. 避免使用 CSS 表达式

CSS 表达式是动态玩 CSS 的一种很强大的方式，但是强大的同时也存在很高的危险性。因为 CSS 表达式的频繁求值会导致 CSS 表达式性能低下。如果真想玩 CSS 表达式，可以选用只求值一次的表达式，或者使用事件处理来改变 CSS 的值。

第 681 道：在使用的开发者工具中，你最喜欢的开发工具是什么？

解析：

据了解，当今前端开发者主要使用的开发工具主要有 Firebug、HttpWatchFiddler、HttpFox、Yslow、cssUsage、VIM、Editplus、DNS Flusher、PageSpeed 这几种，根据自己的习惯来回答即可。

第 682 道：如果今年你打算熟练掌握一项新技术，那会是什么？

解析：

在 JavaScript 越来越流行的今天，Node.js 是一项非常重要的创新。Node.js 是基于 Chrome JavaScript 运行时建立的一个平台，用来搭建快速的、易于扩展的网络应用。

Node.js 借助事件驱动，非阻塞 I/O 模型变得轻便和高效，非常适合分布式设备的数据密集型的实时应用。

V8 引擎执行 JavaScript 的速度非常快，性能非常好。Node.js 对一些特殊用例进行了优化，提供了替代的 API，使得 V8 在非浏览器环境下运行得更好。

第 683 道：能描述一下，你制作一个网页的工作流程吗？

解析：

- (1) 确定网站主题。
- (2) 搜集材料。
- (3) 规划网站。
- (4) 选择合适的制作工具。
- (5) 制作网页。
- (6) 上传测试。
- (7) 推广宣传。
- (8) 更新维护。

第 684 道：前端页面由哪三层构成？分别是什么？作用又是什么？

解析：

网页分为三个层次，即结构层、表现层、行为层。

网页的结构层是由超文本标记语言（HTML）或可扩展超文本标记语言（XHTML）等负责创建的，标签对网页内容的语义做出了解释。

网页的表现层是由 CSS 创建的，作用是对网页内容如何显示进行自定义。

网页的行为层实际上就是运用 JavaScript 等语言对 HTML 文档中所有的元素进行操作，包括美化页面等。

第 685 道：说说最近看过的关于最新的前端技术的书。

解析：

《JavaScript 权威指南（第 6 版）》、《JavaScript 语言精粹》、《基于 MVC 的 JavaScript Web 富应用开发》、《JavaScript 模式》。

第 686 道：前端性能优化知识都有哪些？分别从代码层面和架构方面进行介绍。

解析：

1. 减少 HTTP 请求

基本原理：

在浏览器（客户端）和服务器发生通信时，就已经消耗了大量的时间，尤其是在网络情况比较糟糕时，这个问题尤其突出。

一个正常 HTTP 请求的流程简述：

如在浏览器中输入“www.xxxxxx.com”并按下 Enter 键，浏览器与这个 URL 指向的服务器建立连接，然后浏览器才能向服务器发送请求信息，服务器在接收到请求的信息后，再返回相应的信息，浏览器接收到来自服务器的应答信息后，对这些数据解释执行。

而当我们请求的网页文件中有很多图片、CSS、JS，甚至音乐等信息时，将会频繁地与服务器建立连接和释放连接，这必定会造成资源的浪费，且每个 HTTP 请求都会对服务器和浏览器产生性能负担。

在网速相同的条件下，下载一个 100KB 的图片比下载两个 50KB 的图片要快。所以，请减少 HTTP 请求。

解决办法：

合并图片（CSS Sprites），合并 CSS 和 JS 文件。图片较多的页面也可以使用 lazyLoad 等技术进行优化。

2. 正确理解 Repaint 和 Reflow

【注】Repaint 和 Reflow 就是重绘和重排的意思。

基本原理：

Repaint（重绘）是在一个元素的外观被改变，但没有改变布局（宽和高）的情况下发生的，如改变 visibility、outline、背景色等。

Reflow（重排）就是 DOM 的变化影响到了元素的几何属性（宽和高），浏览器会重新计算元素的几何属性，会使渲染树中受到影响的部分失效，浏览器会验证 DOM 树上的所有其他结点的 visibility 属性，这也是 Reflow 低效的原因。如：改变窗口大小、改变文字大小、内容的改变、浏览器窗口变化、style 属性的改变等。如果 Reflow 得过于频繁，CPU 使用率就会迅速地上升，所以前端也

就有必要知道 Repaint 和 Reflow 的知识。

减少性能影响的办法：

上面提到通过设置 style 属性改变结点样式的话，每设置一次都会导致一次 Reflow，所以最好通过设置 class 的方式。有动画效果的元素，它的 position 属性应当设为 fixed 或 absolute，这样不会影响其他元素的布局。如果在功能需求上不能设置 position 为 fixed 或 absolute，那么就权衡速度的平滑性。

总之，因为 Reflow 有时确实不可避免，所以只能尽可能限制 Reflow 的影响范围。

3. 减少对 DOM 的操作

基本原理：

对 DOM 操作的代价是高昂的，这在网页应用中通常是一个性能瓶颈。

在《高性能 JavaScript》中这么比喻：“把 DOM 看成一个岛屿，把 JavaScript (ECMAScript) 看成另一个岛屿，两者之间以一座收费桥连接”。每次访问 DOM 都会交过桥费，而访问的次数越多，交的费用也就越多。所以一般建议尽量减少过桥次数。

解决办法：

修改和访问 DOM 元素会造成页面的 Repaint 和 Reflow，循环对 DOM 操作更是罪恶的行为，所以请合理地使用 JavaScript 变量储存内容，考虑大量 DOM 元素中循环的性能开销，在循环结束时一次性写入。

减少对 DOM 元素的查询和修改，查询时可将其赋值给局部变量。

【注】在 IE 中，hover 会降低响应速度。

4. 使用 JSON 格式来进行数据交换

基本原理：

JSON 是一种轻量级的数据交换格式，采用完全独立于语言的文本格式，是理想的数据交换格式。同时，JSON 是 JavaScript 原生格式，这意味着在 JavaScript 中处理 JSON 数据不需要任何特殊的 API 或工具包。

与 XML 序列化相比，JSON 序列化后产生的数据一般要比 XML 序列化后产生的数据体积小，所以在 Facebook 等知名网站中都采用了 JSON 作为数据交换方式。

第 687 道：简述您的设计理念或设计风格？

解析：

(1) 功能决定设计方向。网站的功能决定了设计的思路，不同目的网站的设计思路也是不同的。

(2) 内容决定形式。确定了网站的功能之后就需要开始实际的行动，对网站进行整体把握，如内容的填充、细节的划分，以及不同风格的创意效果，用更具吸引力的方式呈现内容是网站表现与众不同的必须要求。

(3) 细节决定成败。一个具有创意的网站设计方案需要注重各个方面的细节，比如说布局、颜色及动态元素的使用。简单明了又清晰的布局是网页设计的基础，毫无章法的网站是没有吸引力的；合适的配色方案是吸引用户眼球的重要因素，不同的色彩搭配给人以不同的视觉冲击力；动态网页技术就是提高界面与用户之间的交互性，使用户体验更加方便和灵活。

第 688 道：对于 IE 的 HTML 元素事件和标准 DOM 的事件模型有什么不同，怎么处理兼容性？

解析：

HTML 元素事件是浏览器内在自动产生的，当有事件发生时，HTML 元素会向外界发出各种 DOM 事件流，并且它只支持事件绑定，不支持事件捕获。在事件处理的函数内部不可以使用 `this` 关键字。

DOM（文档对象模型）结构是一个树形结构，当一个 HTML 触发某一事件时，该事件就会在元素节点与根节点之间的路径传播，路径所经过的节点都会收到该事件。DOM 事件模型的优点是支持事件绑定和事件捕获，并且在事件处理的函数内部可以使用 `this` 关键字。

第 689 道：请至少说出三个常浏览的前端技术相关的网站。

解析：

这个题目考查的是用户经常关注学习技术的来源。推荐三个：cnblogs.com、csdn.net、oschina.com，还有前端江湖（www.qdjh.com）。

第 690 道：请简要阐述你对 MVC 的理解。

解析：

MVC 模式是表示层开发最常用的设计模式，使用这个设计模式的目的是为了解除控制逻辑、业务逻辑和视图之间的耦合，提升系统的可扩展性和可维护性。MVC 模式主要由模型、控制器和视图三个部分组成。其中，控制器的主要责任是接受客户所提交的请求，并将请求转发给适当的模型对象进行处理，再将处理的结果发给视图，进行显示。

第 691 道：请简要阐述你对 REST 的理解。

解析：

REST（REpresentational State Transfer，即表述性状态传递），是一种软件架构风格，提供了一组框架约束，强调组件交互的可伸缩性、接口的通用性、组件的独立部署，以及用来减少交互延迟、增强安全性、封装遗留系统的中间组件。REST 是一套用来创建 Web Service 的方法，它的主要优势在于它是一种对服务器来说更加有效的抽象方式，REST 架构风格是将服务器抽象为一组离散资源的集合。

第 692 道：在移动 Web 开发时，可以使用 `click` 事件吗？如不行，请说明

原因。

解析：

可以使用 click 事件，但是在手机 Web 端，click 会有 200 ~ 300 ms，所以请用 tap 代替 click 作为单击事件。

在移动端选取元素时，推荐使用 `document.qrerySelector("#box");` 和 `document.qrerySelector("#box li")[0];`。

移动端单次触摸事件，直接绑定 touchend 事件即可，所有的手机事件都要阻止默认事件，即 `ev.preventDefault();`。

第 693 道：介绍一下项目的流程。

解析：

首先，介绍项目的背景；其次，介绍流程，以及分工。你从事的是什么，是谁主导这个项目的，你为这个项目做的贡献是什么，你是如何配合同事们完成这个项目的，当出现冲突和困难时你是如何处理的。

第 694 道：怎样优化自己的代码？

解析：

1. 清理垃圾代码

删除页面中的垃圾代码，垃圾代码主要指那些删除了也不会对页面有任何影响的非必要代码。

常见的垃圾代码——空格字符。空格字符是网页中最常见的垃圾代码，但并不是指标签，而是在代码编辑环境下敲击空格所产生的符号，每个空格相当于一个字符，也就是说，一个页面，空格就占页面体积的 15%，即 100KB 的页面，有 15KB 是空格字符。

空格字符最常出现在代码的开始和结束处，还有空行中。这些都是容易产生垃圾代码的地方。消除这样的垃圾代码的方法就是，选中代码，然后按 Shift+Tab 组合键左对齐。

2. CSS 优化

CSS 可以以调用的方式，避免同样的标签重复写样式，从而达到精简代码的效果。

第 695 道：列举几个 Web 2.0 的特征。

解析：

1. 用户参与网站内容制作

与 Web 1.0 网站单项信息发布的模式不同，Web 2.0 网站的内容通常是由用户发布的，用户既是网站内容的浏览者，也是网站内容的制作者，这也就意味着 Web 2.0 网站为用户提供了更多参与的机会，博客网站和 wiki 就是典型的用户创造内容的例子，而 tag 技术（用户设置标签）将传统网站中的信息分类工作直接交给用户来完成。

2. Web 2.0 更加注重交互性

用户在发布内容过程中实现与网络服务器之间交互，即同一网站不同用户之间的交互，以及不同网站之间信息的交互。

3. 符合 Web 标准的网站设计

Web 标准是目前国际上正在推广的网站标准，通常所说的 Web 标准一般是指网站建设采用基于 XHTML 语言的网站设计，实际上，Web 标准并不是某一标准，而是一系列标准的集合。Web 标准中典型的应用模式是“CSS+DIV”，摒弃了 HTML 4.0 中的表格定位方式，其优点之一是网站设计代码规范，并且减少了大量代码，减少网络带宽资源浪费，加快了网站访问速度。更重要的一点是，符合 Web 标准的网站对于用户和搜索引擎更加友好。

4. Web 2.0 与 Web 1.0 没有绝对的界限。

Web 2.0 技术可以成为 Web 1.0 网站的工具，一些在 Web 2.0 概念之前诞生的网站本身也具有 Web 2.0 的特性，例如，B2B 电子商务网站的免费信息发布和网络社区类网站的内容也来源于用户。

5. Web 2.0 的核心不是技术而在于指导思想。

Web 2.0 有一些典型的技术，但技术是为了达到某种目的所采取的手段。Web 2.0 技术本身不是 Web 2.0 网站的核心，重要的在于典型的 Web 2.0 技术体现了具有 Web 2.0 特征的应用模式。因此，与其说 Web 2.0 是互联网技术的创新，不如说是互联网应用指导思想的革命。

第 696 道：你对前端界面工程师这个职位是怎么理解的？它的前景怎么样？

解析：

前端工程师是一个很新的职业，在国内乃至国际上真正开始受到重视的时间不超过 5 年。互联网的发展速度迅猛，网页由 Web 1.0 到 Web 2.0，再到新生的 HTML5、CSS3，到现在手机、4G 网络等新科技的兴起；网页也由最原先的图文为主，到现在各种各样的基于前端技术实现的应用、交互和富媒体的呈现，更多的信息、更丰富的内容、更友好的体验，已经成为网站前端开发的要求，网站的前端开发发生了翻天覆地的变化。网站的开发对前端的需求越来越重要，但目前前端工程师需求远大于供给，前端人才非常紧缺。所以高质量的前端开发工程师将会是后 5 年内一个非常热门的职业，发展的前景非常可观。

第 697 道：简述一下 AppCan。

解析：

AppCan 是正益无线自主研发的国内首个 HTML5 移动应用开发平台，可以简单、快速、高效地开发移动应用。

AppCan 支持跨平台开发，实现一次开发，多平台、多分辨率自动适配，目前已经支持的平台有 iOS、Android、Symbian、Windows Phone。

第 698 道：说一说前端开发中，你遇到的一些问题。

解析:

前端开发中，主流的效果有很多，当然对于自身来讲，也就有很多觉得难的效果，以下列举出了几个比较具有代表性的示例（根据自身情况回答）：

- (1) 关于 JavaScript 跨域请求 Jsonp 的原理。
- (2) 关于 JavaScript 冒泡事件。
- (3) jQuery.attr().prop().data() 区别及全选问题。
- (4) JSDoc 配置和使用。
- (5) HTML5 之 canvas 画布旋转。

第 699 道：你认为自己写过最长的是哪种效果的脚本？

解析:

下面列举出了几种比较好的效果，当然，还需要根据自身情况来回答。

- (1) 原生的焦点图。
- (2) JavaScript 图片的放大缩小。
- (3) 关于 JavaScript 图片延迟加载事件。
- (4) JavaScript 时间冒泡及其排序方法。

第 700 道：说一下 Ajax 的流程。

解析:

- (1) 定义一个全局变量来保存 XMLHttpRequest 对象。

如：var xmlHttp;

- (2) 写一个函数用来创建 XMLHttpRequest 对象。

```
function createXMLHttpRequest() {
    if(window.ActiveXObject) { // 这个是创建一个 IE 浏览器的
XMLHttpRequest 对象;
        xmlHttp=new ActiveXObject('Microsoft.XMLHTTP');
    }else{// 这个是创建一个其他浏览器的 XMLHttpRequest 对象;
        xmlHttp=new XMLHttpRequest();
    }
}
```

- (3) 定义一个回调函数，用于处理你想处理的数据。

```
function handleStateChang() {
    if(xmlHttp.readyState==4) {
        if(xmlHttp.status==200) {
            // 在这里写你要实现的功能
            alert("OK");
        }
    }
}
```

- (4) 定义一个函数，用于与服务器端进行通信。

```
function doSearch(){
    // 创建 XMLHttpRequest 对象;
    createXMLHttpRequest();
    // 将回调函数赋值给 XMLHttpRequest 对象的 onreadystatechange 方法;
    xmlhttp.onreadystatechange=handleStateChang;
    // 调用 XMLHttpRequest 对象的 open 方法，并且给定相关参数
    xmlhttp.open("GET","dynamicContent.xml",true);
    xmlhttp.send(null);
}
```

第 701 道：简述一下复选框的只读效果。

解析：

在 Web 开发中，有时候需要显示一些复选框（checkbox），表明这个地方是可以进行勾选操作的，但是，有时候只是想告知用户这个地方是可以进行勾选操作的，而不想让用户在此处勾选（比如在信息展示页面），这时候就需要将复选框设置成只读的效果。

提到只读，很容易想到使用 readonly 属性，但是对于复选框来说，这个属性和期望得到的效果是有差别的。原因在于 readonly 属性关联的是页面元素的 value 属性（例如 textbox，设置了 readonly 就不能修改输入框的文本内容），而复选框的勾选 / 取消并不改变其 value 属性，改变的只是一个 checked 状态。所以，对于 checkbox 来说，设置了 readonly，仍然是可以勾选 / 取消的。

和 readonly 类似的，还有一个 disabled 属性，这个属性的作用是设置页面元素为不可用，即不可进行任何交互操作（包括不可修改 value 属性、不可修改 checked 状态等）。

第 702 道：JS 使用得多吗，熟练吗？

解析：

需根据自身的情况，具体问题具体分析。

- (1) 一般的 JS 效果还是比较熟练的。
- (2) 用过多种 JS 插件，比如：zepto.JS、exet.JS。

第 703 道：对于 CSS 和 JS，你比较偏向哪个？

解析：

不同的效果偏向不同的技术。

CSS 一般比较倾向于页面的排版效果的实现。JS 则是对于交互效果的实现有着决定性的作用。

第 704 道：怎么与后台衔接？

解析：

- (1) 接收用户请求。
- (2) 找到负责处理的程序。

- (3) 处理程序，找到要传输给用户的前端页面。
- (4) 给前端页面留出位置。
- (5) 后端到数据库取数据。
- (6) 后端把数据放在前端留出来的位置上。
- (7) 结合成真正用户看到的 HTML 文件。
- (8) 传输给用户。

第 705 道：怎么写 jQuery 插件？

解析：

jQuery（以下简称 JQ）是一个功能强大而又小巧的 JS 框架，现在很多网站都在使用 JQ。下面教大家如何写一个属于自己的 JQ 插件。本 JQ 插件的例子是在网站的文章结尾处添加版权信息。JQ 插件标准的封装代码如下。

首先，需要闭包。

```
<script type="text/javascript">
(function ($) {
    // 这里放入插件代码
})(jQuery);
</script>
```

这是 jQuery 官方的插件开发规范，这样写的作用是：

- (1) 避免全局依赖。
- (2) 避免第三方破坏。
- (3) 兼容 jQuery 操作符 '\$' 和 'jQuery'。

接着，给插件加入主体。

```
<script type="text/javascript">
(function ($) {
    $.fn.userCp = function(options) { // 定义插件的名称，这里为 userCp
        var dft = {
            // 以下为该插件的属性及其默认值
            cpBy: "dafi", // 版权所有
            url: "http://www.qdjhu.com", // 所有者链接
            size: "12px", // 版权文字大小
            align: "left" // 版权文字位置, left || center || right
        };
        var ops = $.extend(dft, options);
        var style = 'style="font-size:' + ops.size + ';text-align:' +
            ops.align + '";'; // 调用默认的风格
        var cpTxt = '<p' + ' ' + style + '> 此文章版权归 <a target="_blank" href="' + ops.url + '">' + ops.cpBy + '</a> 所有 </p>';
        // 生成版权文字的代码
        $(this).append(cpTxt); // 把版权文字加入想显示的 DIV
    };
})(jQuery);
```

```

    }
  })(jQuery);
</script>

```

第 706 道：简述 App 效果的流程。

解析：

不论是 iOS 还是 Android 的应用开发，其实都遵循着一定的开发流程，只有这样才能使开发过程有章可循，而不是一团乱。

明确你的构想和理念。

(1) 你的 App 应用是开发给谁用的？是写给小孩玩的游戏，还是用来理财的记账类应用。每个应用都有固定的适用人群，而这决定你应用的内容是什么，也决定了要给使用者以什么样的用户体验。

(2) 你的 App 应用是做什么的？一个 App 有明确的使用目的。如何来明确？一个办法就是要想清楚什么能吸引用户来使用你的 App。

(3) 你的 App 能解决什么问题？一个 App 应该致力于解决好一个问题，而不是处理很多毫无相关的问题，那样你就要考虑开发几个不同的 App。每个 App 都应该专注解决一类问题。

(4) 设计用户界面。有了明确的理念和构想，下面就该设计用户界面了，用户界面是 App 使用者与你的 App 交互的地方，应该遵循简洁、美观、便捷的原则，从 App 使用者的角度出发，带给用户良好的使用体验。

(5) 定义交互方式。

用户界面因为有了交互才变得“活”起来。在 App 开发中，交互的启用都是通过用户操作的事件来触发的，比如手指的单击、滑动、捏合等操作。通过定义这些事件，我们可以精确地对用户的操作给出响应，或者打开新的界面，或者提供展现的内容在当前界面上。

(6) 部署用户行为。

定义好交互方式后，接下来就要通过代码来实现这些定义好的行为了。可以说用户的所有操作都是通过我们实现的定义来得到响应的，如果没有前面明确的定义，那么会给用户带来困扰，这是一个好的 App 所不能出现的错误。

(7) 数据交互的部署。

有了设计好的用户界面和交互方式，接下来就要考虑数据的存储问题。

在界面与数据之间必须要定义明确的交互方式，尽管使用 App 的人不直接和这些数据交互，但是，一个好的数据模型是 App 的坚实基础，它会使你的 App 更有扩展性，更易于将来的修改。

第 707 道：MySQL 能用来干什么？

解析：

MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，目

前属于 Oracle 公司。MySQL 是最流行的关系型数据库管理系统，在 Web 应用方面，MySQL 是最好的 RDBMS（Relational Database Management System，关系数据库管理系统）应用软件之一。

MySQL 是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就提高了速度，并增加了灵活性。MySQL 所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，它分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。由于其社区版的性能卓越，搭配 PHP 和 Apache 可组成良好的开发环境。

第 708 道：你了解 Java 吗？

解析：

了解。Java 是一种可以撰写跨平台应用软件的面向对象的程序设计语言，是由 Sun Microsystems 公司于 1995 年 5 月推出的 Java 程序设计语言和 Java 平台（即 JavaEE、JavaME、JavaSE）的总称。Java 自面世后就非常流行，其发展迅速，对 C++ 语言形成了有力冲击。Java 技术具有卓越的通用性、高效性、平台移植性和安全性，广泛应用于 PC、数据中心、游戏控制台、科学超级计算机、移动电话和互联网，同时拥有全球最大的开发者专业社群。在全球云计算和移动互联网的产业环境下，Java 更具备了显著的优势和广阔的前景。

第 709 道：你知道 Cocos2d 吗？

解析：

Cocos2d 是一个基于 MIT 协议的开源框架，用于构建游戏、应用程序和其他图形界面交互应用。它可以让你在创建自己的多平台游戏时节省很多时间。游戏开发者可以把关注焦点放在游戏设置本身，而不必消耗大量时间学习晦涩难懂的 OpenGL ES，此外，Cocos2d 还提供了大量的规范。Cocos2d 基于 OpenGL ES 进行图形渲染，从而让移动设备的 GPU 性能发挥到极致。另外，2012 年发布的 CocoStudio 工具集是开源游戏引擎 Cocos2d-x 开发团队官方推出的游戏开发工具，目前已经进入稳定期。CocoStudio 吸取了他们在游戏制作中的经验，为移动游戏开发者和团队量身定做，旨在降低游戏开发的门槛，提高开发效率，同时也为 Cocos2d-x 的进一步发展打下基础。

第 710 道：简单说一下 SEO 优化。

解析：

SEO 是指在了解搜索引擎自然排名机制的基础上，对网站进行内部及外部的调整优化，改进网站在搜索引擎中关键词的自然排名，获得更多流量，吸引更多目标客户，从而达到网络营销及品牌建设的目的。

第 711 道：Ajax 底层是依靠什么完成请求的？

解析：

事实上整个 Ajax 的无刷新功能就是利用 XMLHttpRequest 的异步请求来完成的。

首先，我们创建一个 XMLHttpRequest 对象（由于浏览器不同需要相应判断处理），设置相应的请求信息（通过 open 来做，例如请求地址等设置）；

然后，我们绑定 onreadystatechange 事件，在这个事件中，我们根据服务器返回状态的不同来做出不同处理，这其中主要是请求成功后接收相应的返回值，通过 JavaScript 对客户端做出相应操作（我们只是显示有关信息）；

最后，我们发送这个请求（通过 send 方法）。

第 712 道：JS 中的闭包、继承、原型链，你比较熟悉哪个，并简单描述一下。

解析：

1. 闭包

闭包（closure）的概念：闭包，简单的理解，就是能够读取其他函数内部变量的函数；从本质上理解，就是将函数内部和函数外部连接起来的一座桥梁。

闭包的用途：一个是前面提到的可以读取函数内部的变量，另一个就是让这些变量的值始终保持在内存中。

理解 JavaScript 的闭包是迈向高级 JS 程序员的必经之路，理解了其解释和运行机制才能写出更安全和优雅的代码。

2. 继承

JS 中继承跟 Java 中的继承不太一样，一般通过 call() 和 apply() 两种方式完成，JS 中的继承是以复制的形式完成的，复制一个父对象，而不像 Java 中直接继承父对象。通过原型的方式完成继承，也有弊端。总之，JS 中的继承只是形式上的对面向对象语言的一种模仿，本质上不是继承，但用起来效果是一样的。至于为什么要继承，通常在一般的项目里不需要，因为应用简单，但要用纯 JS 做一些复杂的工具或框架系统就要用到了，比如 webgis，或者 JS 框架，如 jQuery、ext 等，不然一个几千行代码的框架不用继承得写几万行，而且无法维护。

3. 原型链

ECMAScript 中描述了原型链的概念。我们知道 ECMAScript 并不像 C++ 和 Java 那样使用类，但是对象仍然可以通过多种方式创建，其中就有构造函数方式。每个构造函数都有一个原型对象，同时都有一个 prototype 属性。prototype 属性指向构造函数的原型对象，它被用来实现基于原型的继承和共享。而原型对象又都默认会取得一个 constructor 属性，这个属性包含一个指向构造函数（prototype 属性所在函数）的指针。每个通过调用构造函数创建的实例对象都拥有一个指向原型对象的指针，ECMA-262 第 5 版中叫这个指针为 [[prototype]]，虽然在脚本上没有标准的方式访问 [[prototype]]，但 Chrome、Firefox 和 Safari

浏览器在每个对象上都支持一个属性 `_proto_`，而在其他实现中，这个属性对脚本是完全不可见的。假如原型对象等于另一个类型的实例，那么它就拥有指向创建该实例的构造函数的原型对象的指针，依此类推，就形成了一条指针链，这就是原型链的概念。

17.14 “助力 14”：综合考察

第 713 道：你了解的世界云计算公司有哪几家？

解析：

2013 年，云计算已经深入实践，国内国外发展趋于同步。整个云计算产业中，IaaS（云的基础设施即服务）、PaaS（平台即服务）和 SaaS（软件即服务）这样的学术性分类的界限日渐模糊。AWS 发布 OpsWorks、Elastic Beanstalk 和 CloudFront 等 PaaS 功能；谷歌在 GAE 之外推出了 GCE；Salesforce.com 集成了 PaaS 产品 Force.com 和 Heroku；Windows Azure 跨 IaaS、PaaS 和 SaaS，要打造私有云、公有云、混合云的统一管理云平台。国内，在梳理了 300 余家创新云计算服务提供商之后，发现融合现状更为明显，云生态系统会取代传统三类划分。其中，云平台提供商和云应用服务提供商渐成生态系统主流，前者寡头，后者分众。前者如阿里云、腾讯开放平台、百度云、新浪、微软 Azure- 世纪互联、盛大云、奇虎 360 开放平台、京东、中国电信、中国移动、中国联通、华为、中兴、品高、Ucloud；后者如蓝汛、瑞星、安全宝、金山云、七牛、酷盘、易思捷、超图软件、八百客、Xtools，以及已在中国市场蓬勃发展了 20 余年的正在转型的 SI（系统集成商）和 ISV（软件开发商）。

17.15 “助力 15”：项目问题

第 714 道：现有一个项目，你的身份是 JS 系统架构师，让你设计一个微博产品，你认为合理的方案是什么？

解析：

1. 微博的介绍

微博是微博客（MicroBlog）的简称，是一个基于用户关系的信息分享、传播，以及获取平台，用户可以通过 Web、WAP，以及各种客户端组件个人社区，以 140 字左右的文字更新信息，并实现即时分享。

微博可以通过手机短信、彩信、WAP、电脑桌面的形式实现个人内容的更新，具有很强的草根性、信息发布的及时性与便捷性。它将便携式的移动信息

终端通过固化的互联网有机地结合起来，使得信息的发布和关注变得更加简单且易于获得。

“微博是希望得到关注的人或企业的一种表达方式”，易观国际 CEO 于扬认为，国内微博的形式已向 Twitter 靠近。

2. 微博的企业用途

由于微博的一些天然特征，使得它对企业的运用更加方面灵活，特别是与传统的企业论坛、留言板、博客等功能对比，更具有优势，且其使用范围也不仅限于此。

(1) 作为信息交互的交流工具，类似论坛等。

(2) 作为企业电子商务网站的产品信息、经验分享的快速通道。如：通过手机拍照产品后，直接通过微博发布出来，询价、寻求帮助或发布产品价格。

(3) 范媒体作用，能够快速获取新闻类信息。

(4) 企业内部管理和沟通的工具，使员工能够向可能对此感兴趣的任何人发布他们的想法、活动，以及可能具有价值的信息。

(5) 由于微博的发布简单、信息量大，对于网站而言能快速提高网站的新信息量，有助于提高搜索引擎的排名。

3. 市场发展情况

目前，市场几大门户网站开始推出微博客，用以帮助网站丰富内容，提供新闻和提升流量。

微软和谷歌已经开始给企业开发微博客系统，用以帮助企业内的交流和沟通。

4. 价格体系

只要支付一定的费用就可以使用属于自己的微博客系统，包括 1 年的互联网空间使用，同时可以绑定自己的专用域名。

第 715 道：请设计一套方案，用于确保页面中 JS 已完全加载。

解析：

```
var n = document.createElement("script");
n.type = "text/javascript";
if(ua.IE){
    n.onreadystatechange = function(){
        var rs = this.readyState;
        if('loaded' === rs || 'complete'===rs){
            n.onreadystatechange = null;
            f(id,url);
        }
    }
}else{
    n.onload = function(){
        f(id,url);
    };
}
```

第 716 道：如何设计突发大规模并发架构？请简单描述一下思路。

解析：

1. HTML 静态化

其实大家都知道，效率最高、消耗最小的就是纯静态化的 HTML 页面，所以我们尽可能使网站上的页面采用静态页面来实现，这个最简单的方法其实也是最有效的方法。但是对于有大量内容并且频繁更新的网站，我们无法全部手动去逐个实现，于是出现了我们常见的信息发布系统——CMS，像我们常访问的各个门户站点的新闻频道，甚至它们的其他频道，都是通过信息发布系统来管理和实现的，信息发布系统可以实现最简单的信息录入，自动生成静态页面，还具备频道管理、权限管理、自动抓取等功能，对于一个大型网站来说，拥有一套高效、可管理的 CMS 是必不可少的。

除了门户和信息发布类型的网站，对于交互性要求很高的社区类型网站来说，尽可能的静态化也是提高性能的必要手段，将社区内的帖子、文章进行实时的静态化。当更新的时候，重新静态化也是大量使用的策略，像 Mop 的大杂烩就是使用了这样的策略，网易社区等也是如此。

同时，HTML 静态化也是某些缓存策略使用的手段，对于系统中频繁使用数据库查询，但是内容更新很小的应用，可以考虑使用 HTML 静态化来实现，比如论坛的公用设置信息，这些信息目前主流的论坛都可以进行后台管理，并且将数据存储数据库中，这些信息其实大量被前台程序调用，但是更新频率很小，可以考虑当后台更新时将这部分内容进行静态化，这样避免了大量的数据库访问请求。

2. 图片服务器分离

大家知道，对于 Web 服务器来说，不管是 Apache、IIS 还是其他容器，图片是最消耗资源的，于是我们有必要将图片与页面进行分离，这是多数大型网站都会采用的策略，他们都有独立的图片服务器，甚至有很多台图片服务器。这样的架构可以降低页面访问请求的服务器系统压力，并且可以保证系统不会因为图片问题而崩溃，在应用服务器和图片服务器上，可以进行不同的配置优化，比如 Apache 在配置 ContentType 时可以尽量少支持，尽可能少的 loadmodule，保证更高的系统消耗和执行效率。

3. 数据库集群和库表散列

大型网站都有复杂的应用，这些应用必须使用数据库，那么，在面对大量访问时，数据库的瓶颈很快就能显现出来，这时，一台数据库将无法很快满足应用，于是我们需要使用数据库集群或者库表散列。

在数据库集群方面，很多数据库都有自己的解决方案，Oracle、Sybase 等都有很好的方案，常用的 MySQL 提供的 Master/Slave 也是类似的方案，您使用了什么样的 DB，就参考相应的解决方案来实施即可。

上面提到的数据库集群由于在架构、成本、扩张性方面都会受到所采用 DB 类型的限制，于是我们需要从应用程序的角度来考虑改善系统架构，库表散列是常用并且最有效的解决方案。将数据库进行分离，不同的模块对应不同的数据库或者表，再按照一定的策略对某个页面或者功能进行更小的数据库散列，比如用户表，按照用户 ID 进行表散列，这样就能够低成本地提升系统的性能，并且有很好的扩展性。搜狐的论坛就是采用了这样的架构，将论坛的用户、设置、帖子等信息进行数据库分离，然后对帖子、用户按照板块和 ID 进行散列数据库和表，最终可以在配置文件中简单的配置便能让系统随时增加一台低成本数据库进来补充系统性能。

4. 缓存

搞技术的都接触过“缓存”一词，很多地方都会用到缓存。网站架构和网站开发中的缓存也是非常重要的，这里先讲述这两种最基本的缓存。高级和分布式的缓存在后面讲述。

架构方面的缓存，对 Apache 比较熟悉的人都能知道，Apache 提供了自己的缓存模块，也可以使用外加的 Squid 模块进行缓存，这两种方式均可以有效地提高 Apache 的访问响应能力。

网站程序开发方面的缓存，Linux 上提供的 Memory Cache 是常用的缓存接口，可以在 Web 开发过程中使用。比如，用 Java 开发时，就可以调用 MemoryCache 对一些数据进行缓存和通信共享，一些大型社区使用了这样的架构。另外，在使用 Web 语言开发时，各种语言都有自己的缓存模块和方法，PHP 有 Pear 的 Cache 模块，Java 就更多了，.NET 不是很熟悉，相信也肯定有。

5. 镜像

镜像是大型网站常采用的提高性能和数据安全性的方法，镜像的技术可以解决不同网络接入商和地域带来的用户访问速度差异，比如 ChinaNet 和 EduNet 之间的差异就促使了很多网站在教育网内搭建镜像站点，使数据进行定时更新或者实时更新。在镜像的细节技术方面，这里不阐述太深，有很多专业的现成的架构和产品可选，也有廉价的通过软件实现的思路，比如 Linux 上的 rsync 等工具。

6. 负载均衡

负载均衡将是大型网站解决高负荷访问和大量并发请求采用的终极解决办法。

负载均衡技术发展了多年，有很多专业的服务提供商和产品可以选择，以下的两个架构可以给大家做参考。

• 硬件四层交换

第四层交换使用第三层和第四层信息包的报头信息，根据应用区间识别业务流，将整个区间段的业务流分配到合适的应用服务器进行处理。第四层交换

功能就像是虚拟 IP，指向物理服务器。它传输业务服从的协议多种多样，有 HTTP、FTP、NFS、Telnet 或其他协议。这些业务在物理服务器基础上，需要复杂的载量平衡算法。在 IP 世界，业务类型由终端 TCP 或 UDP 端口地址来决定，在第四层交换中的应用区间则由源端和终端 IP 地址、TCP 和 UDP 端口共同决定。

在硬件四层交换产品领域，有一些知名的产品可以选择，比如 Alteon、F5 等，这些产品很昂贵，但是物有所值，能够提供非常优秀的性能和很灵活的管理能力。当初，Yahoo 中国有接近 2000 台服务器，使用了三四台 Alteon 就搞定了。

- 软件四层交换

大家知道了硬件四层交换机的原理后，基于 OSI 模型来实现的软件四层交换也就应运而生了，这样的解决方案实现的原理一致，不过性能稍差。但是满足一定量的压力还是游刃有余的，有人说软件实现方式其实更灵活，处理能力完全看你配置的熟悉能力。

我们可以使用 Linux 上常用的 LVS (Linux Virtual Server) 来解决软件四层交换，它提供了基于心跳线 heartbeat 的实时灾难应对解决方案，提高系统的鲁棒性，同时可提供灵活的虚拟 VIP 配置和管理功能，可以满足多种应用需求，这对于分布式的系统来说必不可少。

第 717 道：如果一个项目由你一个人完成，你感觉会遇到什么问题，兼容性怎么解决？

解析：

因为不同浏览器使用内核及所支持的 HTML（标准通用标记语言下的一个应用）等网页语言标准不同，以及用户客户端的环境不同（如分辨率不同），造成的显示效果不能达到理想效果。最常见的问题就是网页元素位置混乱、错位。目前暂没有统一的能解决这样的工具，最普遍的解决办法就是不断地在各浏览器间调试网页显示效果，通过对 CSS 样式控制，以及通过脚本判断并赋予不同浏览器的解析标准。

如果所要实现的效果可以使用框架，那么还有另一个解决办法，即在开发过程中使用当前比较流行的 JS 和 CSS 框架，如 jQuery、YUI 等，因为这些框架无论是底层的还是应用层的一般都已经做好了浏览器兼容，所以可以放心使用。除此之外，CSS 提供了很多 hack 接口可供使用，hack 既可以实现跨浏览器的兼容，也可以实现同一浏览器不同版本的兼容。

17.16 “助力 16”：Flash

第 718 道：如何解决 Flash 挡住层用 z-index 无效的问题？

解析：

在 HTML 中，如果嵌入 Flash，默认会将 Flash 放在页面的最上面。也就是说，即使我们用绝对定位，将 z-index 设得无穷高，页面里的任何元素还是无法将它盖住。这时候，有两种方法可以控制 Flash 的 z-index。

(1) 在嵌入 Flash 时，设置 Flash 的 wmode 参数为 opaque。

(2) 在嵌入 Flash 时，设置 Flash 的 wmode 参数为 transparent。

第 719 道：Flash 可以在 HTML5 中播放吗？

解析：

HTML5 技术相对于 Flash 技术能使用本地播放器播放网页视频（比如 iPhone 不支持 Flash，也能播放网页上的视频），而 Flash 相对于 HTML5 十分吃 CPU，事实上，HTML5 所提供的 API 标准，彻底克制于 Flash 之上。一旦标准普及，毫无悬念，HTML5 将“秒杀”Flash。伴随 HTML5 的普及，以及 Java 性能的逐渐提升，Flash 在前端的领先优势将不复存在，网页低端与高端应用将会产生分流。

第 720 道：怎么样才能让层显示在 Flash 之上？

解析：

可以把 Flash 设置为透明，如下：

```
<a href="http://www.qdljhu.com/">:</a>
<pre lang="html" line="1">
<param name="wmode" value="transparent" />
```

17.17 “助力 17”：逻辑题

第 721 道：甲、乙、丙、丁为四个整数。甲、乙之和与丙、丁之和相等。甲、丁之和大于乙、丙之和。乙大于甲、丙之和。这四个数由大到小的顺序应该是（ ）

A. 甲、乙、丙、丁

B. 丁、丙、乙、甲

C. 丙、乙、丁、甲

D. 丁、乙、甲、丙

答案：D

解析：

甲 + 乙 > 丙 + 丁 ①

甲 + 丁 > 乙 + 丙 ②

乙 + 丁 > 甲 + 丙 ③

①②相加得到甲 > 丙

①③相加得到乙 > 丙

②③相加得到丁 > 丙

由以上的推论来看，丙最小。

第 722 道：刘健、马明、张益三个男生各有一个妹妹，这天，六个人一起打乒乓球，规定是男女混合双打，并且兄妹两人不搭伴。

第一盘对局情况是：刘健和小萍 VS 张益和小英。

第二盘对局情况是：张益和小红 VS 刘健和马明的妹妹。

请根据题干的条件，判断以下哪项为真？

- A. 刘健和小红、马明和小英、张益和小萍各是兄妹。
- B. 刘健和小英、马明和小萍、张益和小红各是兄妹。
- C. 刘健和小萍、马明和小英、张益和小红各是兄妹。
- D. 刘健和小红、马明和小萍、张益和小英各是兄妹。

答案：A

解析：

由题干可知张益和小英不是兄妹，所以张益只能和小萍是兄妹，所以选项 A 正确。

第 723 道：有两根不均匀分布的香，烧完每根香的时间是一个小时，你能用什么方法来确定一段 15 分钟的时间？

解析：

第一根香从两端同时燃烧，同时点燃第二根香的一端，此时记为时间点 A。

当第一根香烧完时为半个小时，这时点燃第二根香的另一端，此时记为时间点 B。

当第二根香烧完时，此时记为时间点 C。

时间点 B 与时间点 C 之间为 15 分钟。

第 724 道：没有人和每一个人都是朋友，张兰和李卉是朋友，李卉和李兰的每一个朋友都是朋友。如果以上陈述为真，则下列哪项不可能为真？

- (1) 每个人和张兰都是朋友。
- (2) 每一个人都和某一些人是朋友。
- (3) 存在某一个人，他和任何人都不是朋友。

- A. 仅 1 B. 仅 2 C. 仅 3 D. 1 和 2

答案：A

解析：

“每个人和张兰都是朋友”表明“有人和每一个人都是朋友”；既然“没有人和每一个人都是朋友”，那当然就不可能存在某一个人和每一个人都是朋友，所以 (1) 不可能为真。“没有人和每一个人都是朋友”并不排斥每一个人都可以和某一些人是朋友，所以 (2) 不一者定为假。“存在某一个人，他和任何人都不是朋友”和“不存在某一个人，他和每一个人都是朋友”并不相互排斥，所以 (3) 可能为真。

第 725 道：某地有一个村庄，村庄里住着骑士和无赖两种人。骑士总是讲真话，无赖总是讲假话。一天，一位了解这一情况的学者路过这个村庄，看见

该村 A、B 两个人，他向 A 提出了一个问题：“你俩中有骑士吗？”A 回答说：“没有”。学者听了 A 的回答，想了一想，就正确地推出了 A 和 B 各是哪种人。假设学者的推断是正确的，以下哪项是学者作出的判断？

- A. A 是骑士，B 是无赖 B. A 和 B 都是骑士
C. A 和 B 都是无赖 D. A 是无赖，B 是骑士

答案：D

解析：

假设 A 是骑士，则事实上两人中有骑士，因而他的回答就是句假话，但骑士不会说假话，因此，假设不成立，A 不是骑士，而是无赖。由于无赖只讲假话，因此 A 的回答是句假话，即事实上两人中有骑士。因为 A 不是骑士，因此，B 一定是骑士。所以，A 是无赖，B 是骑士。

第 726 道：有排列成一行的 4 户人家，A 家在 B 家的隔壁，A 家与 D 家不相邻，D 家与 C 家不相邻，

请问：C 家的隔壁是哪家（ ）

答案：A 家或者 D 家

解析：

1. A 家在 B 家的隔壁，可得 AB 或 BA。
 2. A 家与 D 家不是邻居，可得 ABCD，或 ABDC，或 CABD，或 CDBA，或 DCBA，或 DBAC。
 3. 如果 D 家与 C 家也不相邻，可得 CABD，或 DBAC。
- 故此，C 家的邻居可能是 A 或者 D。
4. 有排列成一行的四户人家，所以不能组成环形，D 排除。

第 727 道：假如半夜 12 点钟开始下雨，那么 72 小时后能出太阳吗？

- A. 能 B. 不能 C. 不知道

答案：B

解析：

根据实际情况分析，72 小时后还是半夜 12 点，那么当然不会出太阳。

第 728 道：一堆西瓜，一半的一半的一半比一半的一半少半个，这堆西瓜有多少个？

解析：

由题意可列方程

$$1/4x - 1/8x = 0.5$$

$$1/8x = 0.5$$

$$x = 4$$

应该是 4 个。

很简单，就是比它多切一半的少了二分之一一个西瓜。所以说少切了一半那

个就必须是 1 个西瓜了。

第 729 道：数学竞赛共有 10 题，每答对一题得 5 分，不回答或答错一题扣 3 分，要得到 34 分必须答对多少题？

答案：要得到 34 分必须答对 8 题。

解析：

假设你答对 x 道题，不回答或者答错就是 $(10-x)$ 道题，则：

$$5x - 3(10 - x) = 34$$

$$5x - 30 + 3x = 34$$

$$8x = 64$$

$$x = 8$$

第 730 道：你有两个桶，容量分别是 3 升和 5 升，同时还有大量的水，你怎么才能准确量出 4 升的水？

解析：

- (1) 将 5 升桶倒满。
- (2) 把 5 升桶的水倒满 3 升桶，剩下 2 升水。
- (3) 将 3 升桶的水倒光。
- (4) 把 5 升桶里剩下的 2 升水倒入 3 升桶。
- (5) 将 5 升桶倒满。
- (6) 把 5 升桶的水倒入 3 升桶至满。
- (7) 此时 5 升桶里刚好剩下 4 升水。

第 731 道：你有四个装药丸的罐子，每罐内有多药丸，其中一个罐的药丸全被污染，干净药丸都是 1g，被污染的药丸是 2g，只称量一次，如何判断哪个罐子的药丸被污染了？

解析：

第一个药罐拿 1 个药丸；第二个药罐拿 2 个药丸；第三个药罐拿 3 个药丸；第四个药罐拿 4 个药丸。开始计算标准的 10 颗药重量，并且与现在的 10 颗药比较：如果重量多 1g，就是第一个药罐污染了；如果重量多 2g，就是第二个药罐污染了；如果重量多 3g，就是第三个药罐污染了；如果重量多 4g，就是第四个药罐污染了。

第 732 道：若一棵树的其中一面布满青苔，这样的青苔怎样分南北呢？

解析：

由常识可知：长在石头上的青苔性喜潮湿，不耐阳光，因而青苔通常生长在石头的北面。

第 733 道：请填写此数列中下一个数：6,10,14,18,22, ____。

解析：

由规律可得：第 n 个数是 $6 + 4(n-1) = 4n + 2$ 。

第二个是第一个加 4，第三个是第一个加两个 4，第四个是第一个加三个

4……第 n 个是第一个加 $n-1$ 个 4。

填空处为第六个数，所以该处应为 $6+4(6-1)=4\times 6+2=26$ 。

第 734 道：下列选项中，哪一个应该填在“XOOOOXXOOOXXX”后面（ ）。

A. XOO B. OOX C. OXO D. OXX

答案：B

解析：

观察“XOOOOXXOOOXXX”可知：

顺序为一个 X、两个 X、三个 X，然后是四个 O、三个 O，所以后面应为两个 O。

第 735 道：括号里应填什么字母？B、F、K、Q、（ ）

答案：T

解析：

B、F、K、Q 在 26 个英文字母表中对应位置为 2、6、11、17，该数列是一个二级等差数列，一级差为 4、5、6，二级差为 1，即下个字母在字母表中的对应位置应是 $6+1+13=20$ ，即字母 T。

第 736 道：1 元钱 1 瓶汽水，喝完后 2 个空瓶换 1 瓶汽水。问：你有 20 元钱，最多可以喝到几瓶汽水？

解析：

对于这道题有以下三种解题思路。

解题思路 1：

一开始 20 瓶没有问题，随后的 10 瓶和 5 瓶也都没有问题，接着把 5 瓶分成 4 瓶和 1 瓶，前 4 个空瓶再换 2 瓶，喝完后 2 瓶再换 1 瓶，此时喝完后手头上剩余的空瓶数为 2 个，把这 2 个瓶换 1 瓶继续喝，喝完后把这 1 个空瓶换 1 瓶汽水，喝完换来的那瓶再把瓶子还给人家即可，所以最多可以喝的汽水数为： $20+10+5+2+1+1+1=40$ 。

解题思路 2：

先看 1 元钱最多能喝几瓶汽水。喝 1 瓶余 1 个空瓶，借商家 1 个空瓶，2 个瓶换 1 瓶继续喝，喝完后把这 1 个空瓶还给商家。即 1 元钱最多能喝 2 瓶汽水。20 元钱当然最多能喝 40 瓶汽水。

解题思路 3：

2 个空瓶换 1 瓶汽水，可知纯汽水只值 5 角钱。20 元钱当然最多能喝 40 瓶的纯汽水。N 元钱当然最多能喝 2N 瓶汽水。

由以上可知，假如我有 20 元，最多可以喝到 40 瓶汽水。

第 737 道：一群人开舞会，每人头上都戴着一顶帽子。帽子只有黑白两种，黑的至少有一顶。每个人都能看到其他人帽子的颜色，却看不到自己的。主持

人先让大家看看别人头上戴的是什么颜色的帽子，然后关灯，如果有人认为自己戴的是黑帽子，就打自己一个耳光。第一次关灯，没有声音。于是再开灯，大家再看一遍，关灯时仍然鸦雀无声。一直到第三次关灯，才有劈劈啪啪打耳光的声音响起。问有多少人戴着黑帽子？

解析：

有三个人戴黑帽子。假设有 N 个人戴黑帽子，当 $N=1$ 时，戴黑帽子的人看见别人都为白则能肯定自己为黑。于是第一次关灯就应该有声。可以断定 $N>1$ 。对于每个戴黑帽子的人来说，他能看见 $N-1$ 顶黑帽，并由此假定自己为白。但等待 $N-1$ 次还没有人打自己以后，每个戴黑帽子的人都能知道自己也是黑的了。所以第 N 次关灯就有 N 个人打自己。

第 738 道：有三个人去住旅馆，住三间房，每一间房 10 元，于是他们一共付给老板 30 元，第二天，老板觉得三间房只需要 25 元就够了，于是叫服务员退回 5 元给三位客人。谁知服务员贪心，给每人只退回 1 元，自己偷偷拿了 2 元，这样一来便等于那三位客人每人各花了 9 元，于是三个人一共花了 27 元，再加上服务员独吞了的 2 元，总共是 29 元。可是当初他们三个人一共支付出 30 元，那么还有 1 元呢？

解析：

这是个典型的偷梁换柱的题目。问题应为“服务员拿了 2 块钱”，是问者输入错误。

首先，我们应该弄清楚“他们每个人出了 9 块钱”是怎么回事：

三人交： $10 \times 3 = 30$

老板收： $30 - 5 = 25$

每人给老板： $25 \div 3 = 25/3$

每人被服务员拿： $2 \div 3 = 2/3$

每人给老板和服务员的总额： $25/3 + 2/3 = 9$ ，每人给出 10 元，每人给老板和服务员的总额 9 元，所以应被找回 1 元。

其次，我们应该弄清楚为什么会出现“还有 1 元去哪里了”的现象：三人给老板和服务员的总额： $25 + 2 = 27$ ，即“每个人出了 9 元”， $9 \times 3 = 27$ ，这 27 元包括服务员拿的 2 元了。

题目中有意在已经包含了“被服务员拿去的 2 元”的 27 元上，又加了一次“被服务员拿去的 2 元”来麻痹大家，却没有加应该“找回的 3 元”。其实，我说“他们每个人出了 9 元，服务员拿的 2 元包括在这三个 9 元里了”，你是不是就已经明白了呢？也就是说，本来应该是： $(10 \times 3 - 5) + 2 + 3 = 30$ ，却被算作了： $(10 \times 3 - 5) + 2 + 2 = 29$ 。

第6篇

功成时

综合测试莫言愁

，把酒笑

综合测试把重要的概念以知识链接的形式呈现，让我们不再为面试题而发愁。笔者通过大量资料查询和整理工作，并且根据实时的企业现状，编写了几套面试模拟题，相信会帮助大家更上一层楼。

第 18 章



前端开发面试题

18.1 前端面试模拟试题一

一、技术题

JS 题型

1. JavaScript 是一门什么样的语言，它有哪些特点？

答：JavaScript 是一门脚本编写语言。它的特点是具有简单性、安全性、动态性、跨平台性。（此题没有标准答案，仅供参考）

2. JavaScript 的数据类型都有什么？

答：字符串、数字、布尔、数组、对象、Null、Undefined。

3. 已知 ID 的 Input 输入框，希望获取这个输入框的值，怎么做？（不使用第三方框架）

答：`document.getElementById("ID").value;`

4. 希望获取到页面中所有的 checkbox 怎么做？（不使用第三方框架）

```
答：var list=document.getElementsByTagName('input');
    var check=[];
    var len=list.length;  // 缓存到局部变量
    while(len--){
        if(list[len].type=='checkbox'){
            check.push(list[len]);
        }
    }
```

5. 设置一个已知 ID 的 DIV 的 HTML 内容为 xxxx，字体颜色设置为黑色，请写出代码。（不使用第三方框架）。

```
答：var cont=document.getElementById("ID");
    cont.innerHTML="xxxx";
    cont.style.color="black"
```

6. 当一个 DOM 节点被单击时，我们希望能够执行一个函数，应该怎么做？

答：首先，在 DOM 里绑定事件：`<div onclick="text()"></div>`

然后，在 JS 中通过 onclick 绑定：xxx.onclick=text

最后，通过事件添加进行绑定：addEventListener(xxx, "click", test)

7. 什么是 Ajax 和 JSON？它们各有什么优缺点？

答：Ajax 指异步 JavaScript 及 XML（Asynchronous JavaScript And XML）。

优点：

(1) 最大的优点是页面无刷新，用户的体验非常好。

(2) 使用异步方式与服务器通信，具有更加迅速的响应能力。

(3) 可以把以前一些服务器负担的工作转嫁到客户端，利用客户端闲置的能力来处理，减轻服务器和带宽的负担，节约空间和宽带租用成本，并且减轻服务器的负担。Ajax 的原则是“按需取数据”，可以最大程度地减少冗余请求和响应应对服务器造成的负担。

(4) 基于标准化的并被广泛支持的技术，不需要下载插件或者小程序。

缺点：

(1) Ajax 不支持浏览器 back 按钮。

(2) 安全问题。Ajax 暴露了与服务器交互的细节。

(3) 对搜索引擎的支持比较弱。

(4) 破坏了程序的异常机制。

(5) 不容易调试。

JSON 是 JavaScript 对象表示法（JavaScript Object Notation），是存储和交换文本信息的语法，类似 XML。它比 XML 更小、更快、更易解析。

优点：

(1) 作为一种数据传输格式，JSON 与 XML 很相似，但是它更加灵巧。

(2) JSON 不需要从服务器端发送含有特定内容类型的首部信息。

缺点：

(1) 语法过于严谨。

(2) 代码不易读。

(3) eval 函数存在风险

8. 下列代码输出什么？解释原因。

```
1 var a;2 alert(typeof a); // undefined3 alert(b); // 报错
```

答：undefined 是一个数据类型，只有一个值，这个值就是“undefined”，在使用 var 声明变量时，并没有对其赋值，这个变量的值就是 undefined，而 b 由于未声明将报错。

9. 下列代码输出什么？解释原因。

```
1 var a = null;2 alert(typeof a); //object
```

答：null 是一个数据类型，只有一个值，这个值就是 null，表示一个无类型指针，所以用 typeof 检测会返回“object”。

10. 下列代码输出什么？解释原因。

```
1 var undefined;2 undefined == null; // true3 1 == true; // true4 2 == true; // false5 0 == false; // true6 0 == ""; // true7 NaN == NaN; // false8 [] == false; // true9 [] == ![]; // true
```

答：undefined 与 null 相等，但不恒等（===）。当一个数是数，另一个是数组时，会尝试将数组转换为数。如何将 object 转换为 number 或 string，取决于另外一个对比量的类型，对于 0、空字符串的判断，建议使用“===”。“===”会先判断两边的值类型，类型不匹配时，值为 false。

11. 看代码给答案。

```
1 var a = new Object();2 a.value = 1;3 b = a;4 b.value = 2;5 alert(a.value);
```

答：2

12. 已知数组 var stringArray = ["This", "is", "Baidu", "Campus"], alert "This is Baidu Campus"

答：alert(stringArray.join(""))

本题考察基础 API。

13. var numberArray = [3,6,2,4,1,5];

(1) 实现对该数组的倒排，输出 [5,1,4,2,6,3]

(2) 实现对该数组的降序排列，输出 [6,5,4,3,2,1]

答：function num(sort){
var arr=sort.split("-");
for(var i=1;i<arr.length;i++){
arr[i]=arr[i].CharAt(0).toUpperCase()+arr[i].substr(1,arr[i].length)
sort=arr.join("-");
return sort;
}
}

14. 输出今天的日期，以 YYYY-MM-DD 的方式，比如今天是 2014 年 9 月 26 日，则输出 2014-09-26。

答：var d= new Date();// 获取年 getFullYear() 返回 4 位的数字
var year=d.getFullYear();// 获取月，月份比较特殊，0 是 1 月，11 是 12 月
var month=d.getMonth()+1;// 变成两位
month=month<10 ? '0' +month:month;// 获取日
var day<10 ? '0' +day:day;
alert(year+ '-' +month + '-' +day);

15. 将字符串 "<tr><td>{\$id}</td><td>{\$name}</td></tr>" 中的 {\$id} 替换成 10，{\$name} 替换成 Tony。（使用正则表达式）

答："<tr><td>{\$id}</td><td>{\$id}_{\$name}</td></tr>".replace(/{\$id}/g,

'10').replace(/\{\$name\}/g, 'Tony')

16. 为了保证页面输出安全，我们经常需要对一些特殊的字符进行转义，请写一个函数 escapehtml，将 <, >, &，“进行转义。

答：function escapehtml(str){
 return str.repalce(/[<>"&]/g,function(math){
 switch (math){
 Case "<" :
 return "<" ;
 case ">" ;
 return ">" ;
 case "&" ;
 return "&" ;
 case "\" ;
 return "\" ;
 return "" ;
 }
 })
 }

17. foo = foo && bar，这行代码是什么意思？为什么要这样写？

答：if(!foo) foo = bar;

短路表达式：作为“&&”和“||”操作符的操作数表达式，在进行求值时，最终的结果就可以确定真假，求值过程结束。

18. 看下列代码，将会输出什么？（变量声明提升）

```
var foo = 1;
function(){
    console.log(foo);
    var foo = 2;
    console.log(foo);
}
```

答：输出 undefined 和 2。

19. 用JS实现随机选取 10 ~ 100 之间的 10 个数字，存入一个数组，并排序。

答：function getRandom(istart,iend){// 列出需要被选中的数据

```
var a1=[];
for(var i=0;i<91;i++){
    a1[i]=i+10;
}
// 挑选出随机数据
var a2=[];
for(var j=0;j<10;j++){
    a2.push(a1.splice(Math.floor(Math.randow()*a1.length),1));
    a2.sort();
}
```

```

        alert(a2)
    }

    getRandomow(10,100);

```

20. 把两个数组合并，并删除第二个元素。

答: `var array1 = ['a','b','c'];`
`var array2= ['d','e','f'];`
`var array3 = array1.concat(array2);`
`array3.splice(1,1);`

21. 怎样添加、移除、移动、复制、创建和查找节点（原生 JS，没细化每一步）

答: (1) 添加、移除、替换、插入。

```

appendChild() // 添加
removeChild() // 移除
replaceChild() // 替换
insertBefore() // 插入

```

(2) 创建新节点。

```

createDocumentFragment() // 创建一个 DOM 片段
createElement() // 创建一个具体的元素
createTextNode() // 创建一个文本节点

```

(3) 查找。

```

getElementByTagName() // 通过标签名称
getElementByName() // 通过元素的 name 属性的值
getElementById() // 通过元素的 id, 唯一性

```

22. 正则表达式构造函数 `var reg=new RegExp("xxx")` 与正则表达式字面量 `var reg=` 有什么不同？如何匹配邮箱的正则表达式？

答：当使用 `RegExp()` 构造函数时，不仅需要转义引导（\），而且需要双反斜杠（\\）。

匹配邮箱的正则表达式：

```

/^([a-z]([a-z0-9]*[-_]?[a-z0-9]+)*@[a-z0-9]*[-_]?[a-z0-9]+)+[\\.]([a-z]{2,3}([\\.]
[a-z]{2})?)$/i;

```

23. 看下面代码，给出输出结果。

```

for(var i=1;i<=3;i++){
    setTimeout(function(){
        console.log(i);
    },0);
};

```

答: 4 4 4

24. 写一个 function，清除字符串前后的空格。（兼容所有浏览器）

答: `function trim(mystr) {`
`while ((mystr.indexOf(" ") == 0) && (mystr.length > 1)) {`
`mystr = mystr.substring(1, mystr.length);`


```

    } // 去除前面空格
    while ((mystr.lastIndexOf(" ") == mystr.length - 1)
        && (mystr.length > 1)) {
        mystr = mystr.substring(0, mystr.length - 1);
    } // 去除后面空格
    if (mystr == " ") {
        mystr = "";
    }
    return mystr;
}

```

25. JavaScript 中 callee 和 caller 有何作用？

答：callee 是返回正在被执行的 function 函数，也就是所指定的 function 对象的正文。caller 返回一个对函数的引用，即调用了当前函数的函数体。

26. 实现一个函数 clone，可以对 JavaScript 中的 5 种主要的数据类型（包括 Number、String、Object、Array、Boolean）进行值的复制。

考察点 1：对于基本数据类型和引用数据类型在内存中存放的是值还是指针这一区别是否清楚。

考察点 2：是否知道如何判断一个变量是什么类型的。

考察点 3：递归算法的设计。

答：Object.prototype.clone=function(){
 var o=this.constructor===array ? []:{};
 for(var e in this){
 o[e]=typeof this[e]=== "object" ? This[e].clone :this[e];
 }
 return o;
}

27. 如何消除一个数组里面重复的元素？

答：var arr=[1,2,3,3,4,4,5,5,6,1,9,3,25,4];

```

function deRepeat(){
    var newArr=[];
    var obj={};
    var index=0;
    var l=arr.length;
    for(var i=0;i<l;i++){
        if(obj[arr[i]]==undefined){
            obj[arr[i]]=1;
            newArr[index++]=arr[i];
        }else if(obj[arr[i]]==1)
            continue;
    }
    return newArr;
}

```

```

    }
    var newArr2=deRepeat(arr);
    alert(newArr2);

```

28. 小贤是一条可爱的小狗 (Dog)，它的叫声很好听 (wow)，每次看到主人时就会乖乖叫一声 (yelp)。从这段描述可以得到以下对象：

```

function Dog() {
    this.wow = function() {
        alert(' Wow' );
    }
    this.yelp = function() {
        this.wow();
    }
}

```

小芒和小贤一样，原来也是一条可爱的小狗，可是突然有一天疯了 (MadDog)，看到人就会每隔半秒叫一声 (wow)，且不停地叫唤 (yelp)。请根据描述，按示例的形式用代码来实现。(继承，原型，setInterval)

答：

```

function MadDog(){
    this yelp=function(){
        var self=this;
        setInterval(function(){
            self.wow(
            },500)
        }
    }
    MadDog.prototype=new Dog();
    var dog=new Dog();
    Dog.yelp();
    var madDog=new MadDog();
    madDog.yelp();
}

```

29. 下面这个 ul，如何单击每一列时 alert 其 index (闭包)？

<ul id=" test" >2 这是第一条 3 这是第二条 4 这是第三条 5

答：

```

var lis=document.getElementById( '2223' ). getElementByTagName
( 'li' );
for(var i=0;i<3;i++){
    lis[i].index=i;
    lis[i].onclick=function(){
        alert(this.index);
    };
}

```

30. 编写一个 JavaScript 函数，输入指定类型的选择器（仅需支持 id、class、tagName 三种简单 CSS 选择器，无需兼容组合选择器）可以返回匹配的 DOM 节点，需考虑浏览器兼容性和性能。

```

答: var query = function(selector) {
    var reg = /^(#)?(\.)?(\w+)$/img;
    var regResult = reg.exec(selector);
    var result = [];
    // 如果是 id 选择器
    if(regResult[1]) {
        if(regResult[3]) {
            if(typeof document.querySelector === "function") {
                result.push(document.querySelector(regResult[3]));
            }
            else {
                result.push(document.getElementById(regResult[3]));
            }
        }
    }
    // 如果是 class 选择器
    else if(regResult[2]) {
        if(regResult[3]) {
            if(typeof document.getElementsByClassName === 'function')
            {
                var doms = document.getElementsByClassName(regResult[3])
                if(doms) {
                    result = converToArray(doms);
                }
            }
            // 如果不支持 getElementsByClassName 函数
            else {
                var allDoms = document.getElementsByTagName("*") ;
                for(var i = 0, len = allDoms.length; i < len; i++) {
                    if(allDoms[i].className.search(new
                    RegExp(regResult[2])) > -1) {
                        result.push(allDoms[i]);
                    }
                }
            }
        }
    }
    // 如果是标签选择器
    else if(regResult[3]) {

```

```

        var doms = document.getElementsByTagName(regResult[3].
toLower Case());
        if(doms) {
            result = converToArray(doms);
        }
    }
    return result;
}
function converToArray(nodes){
    var array = null;
    try{
        array = Array.prototype.slice.call(nodes,0);// 针对非
        IE 浏览器
    }catch(ex){
        array = new Array();
        for( var i = 0 ,len = nodes.length; i < len ; i++ ) {
            array.push(nodes[i])
        }
    }
    return array;
}

```

31. 请评价以下代码，并给出改进意见。

```

if(window.addEventListener){
    Var addListener=function(el,type,listener,useCapture){
        el.addEventListener(type,listener,useCapture);};
}else if(document.all){
    addListener = function(el,type,listener){
        el.attachEvent("on"+type,function(){listener.apply(el);10 });
    }
}

```

答：

- (1) 不应该在 if 和 else 语句中直接使用 addListener 函数，应该先声明；
- (2) 不需要使用 window.addEventListener 或 document.all 来检测浏览器，应该使用能力检测。因为 attachEvent 在 IE 中有 this 指向问题，所以不可以直接调用。

改进意见：

```

function addEvent(elem, type, handler){
    if(elem.addEventListener){
        elem.addEventListener(type, handler, false);
    }else if(elem.attachEvent){
        elem['temp' + type + handler] =handler;
        elem[type + handler] = function(){

```

```

    elem['temp' + type + handler].apply(elem);    };
    elem.attachEvent('on' + type, elem[type + handler]);
}
else{
    elem['on' + type] = handler;
} }

```

32. 给 String 对象添加一个方法，传入一个 String 类型的参数，然后将 String 的每个字符间添加空格返回，例如：

```

addSpace("hello world") // -> 'hello  world'
String.prototype.spacify = function(){
    return this.split('').join(' ');
};

```

(1) 直接在对象的原型上添加方法是否安全？尤其是在 Object 对象上。

答：不安全可能在项目工作中被其他框架或库的方法给覆盖掉。

(2) 函数声明与函数表达式有什么区别？

答：事实上，JS 的解析器对函数声明与函数表达式并不是一视同仁地对待的。对于函数声明，JS 解析器会优先读取，确保在所有代码执行之前声明已经被解析，而函数表达式，如同定义其他基本类型的变量一样，只在执行到某一句时才会对其进行解析，所以在实际中，它们还是有差异的，具体表现在，当使用函数声明的形式来定义函数时，可将调用语句写在函数声明之前，而后者这样做的话会报错。

33. 定义一个 log 方法，让它可以代理 console.log 的方法。

答：function log(text) {
 console.log(text);
}

```

log("hello world!")

```

34. 在 JavaScript 中，什么是伪数组？如何将伪数组转化为标准数组？

答：伪数组能通过 Array.prototype.slice 转换为真正的数组，它是带有 length 属性的对象。这种对象有很多，比较特别的是 arguments 对象，还有像调用 getElementsByTagName, document.childNodes 之类的，它们都返回 NodeList 对象，都属于伪数组。我们可以通过 Array.prototype.slice.call(fakeArray) 将伪数组转换为真正的 Array 对象。

35. 对作用域上下文和 this 的理解，看下列代码：

```

var User = {
    count: 1,
    getCount: function() {
        return this.count;
    }
};

```

```
console.log(User.getCount());
var func = User.getCount;
console.log(func());
```

问两处 console 输出什么？为什么？

答：输出 1 和 undefined。

因为 function 是在 window 的上下文中被执行的，所以访问不到 count 属性。

36. 原生 JS 的 window.onload 与 jQuery 的 \$(document).ready(function){} 有什么不同？jQuery 中的 ready() 方法使用 JS 是怎么实现的？

答：window.onload() 方法是必须等到页面内的所有元素（包括图片）加载完毕后才能执行。

\$(document).ready() 在 DOM 结构绘制完毕后就执行，不必等到加载完毕。

```
function ready(fn){
    if(document.addEventListener){
        document.addEventListener('DOMContentLoaded', function() {
            document.removeEventListener('DOMContentLoaded',arguments.callee,
            false);
            fn();
        }, false);
    }else if(document.attachEvent) {
        document.attachEvent('onreadystatechange', function() {
            if(document.readyState == 'complete') {
                document.detachEvent('onreadystatechange', arguments.callee);
                fn();
            }
        });
    }
};
```

37.（设计题）如何实现一个对页面某个节点的拖曳？（使用原生 JS）

答：

（1）需要三个主要事件：mousedown 鼠标按下事件、mousemove 鼠标移动事件、mouseup 鼠标按键抬起事件。

（2）触发 mousedown 事件后，开始拖曳。

（3）mousemove 时，需要通过 event.clientX 和 clientY 获取拖曳位置。

（4）mouseup 时，拖曳结束。

38. 用面向对象的 JavaScript 来介绍一下自己。

答：function extend(sup, overrides) {
 var sub = overrides && overrides.constructor || function() {
 sup.apply(this, arguments);
 };
 sub.prototype = sup.prototype;
 sub.prototype.constructor = sub;
 return sub;
};

```

    var fn = function() {};
    var subp;
    fn.prototype = new sup;
    subp = sub.prototype = fn.prototype;
    subp.constructor = sub;
    apply(subp, overrides);
    sub.superClass = sup.prototype;
    return sub;
  }

  function apply(o, c) {
    if (o && c && typeof c == 'object') {
      for (var p in c) {
        o[p] = c[p];
      }
    }
  }

  return o;
}

function Frontender() {
  this.skills = ['HTML', 'CSS', 'JavaScript'];
}

apply(Frontender.prototype, {
  code: function() {
    console.log('我能编写和处理业务逻辑代码');
  },
  paint: function() {
    console.log('我能还原设计稿');
  },
  debug: function() {
    console.log('我能使用工具调试代码');
  }
});

var Me = extend(Frontender, {
  constructor: function() {
    var _character = '';
    return function Me(config) {
      config = config || {};
      Me.superClass.constructor.call(this);
      this.skills = this.skills.concat(['HTML5', 'CSS3',
'jQuery', 'Zepto', 'jQuery', 'Ajax']);
      apply(this, config);
    }
  }
})();

```

```

    paint: function() {
        Me.superClass.paint.call(this);
        console.log(' 我可以CSS3、HTML5 还原设计稿 ');
    },
    code: function() {
        Me.superClass.code.call(this);
        console.log(function() {
            }.toString().split('\n').slice(2, -2).toString().
replace(/\t/g, ' ').split(',').join('\n'))
        },
    manage: function() {
        console.log(' 我有一个属于自己的代码库和记录知识点的博客 ');
    },
    habit: function() {
        console.log('twitter、HTML5Rock、CSS3-trick、W3.org...');
    },
    enjoy: function() {
        console.log(' 我喜欢打篮球和看 NBA，最喜欢的球员是 Steve•Nash');
    }
});
var me = new Me({
    ChineseName: 'xxx',
    EnglishName: 'xxx',
    Birthday: 'xxxx',
    Email: 'xxx.com',
    School: 'xxx'
});
console.dir(me);

```

39. 讲解用原生 JS 实现 Ajax 的原理。

答：Ajax 是异步通信的一种技术，主要实现技术是 JavaScript+XML+HTML+ CSS+ 服务端。Ajax 的技术核心是 XMLHttpRequest 对象；Ajax 请求过程：创建 XMLHttpRequest 对象、连接服务器、发送请求、接收响应数据。例如：

```

Ajax({url: "../TestXHR.aspx", // 请求地址
type: "POST", // 请求方式
data: { name: "super", age: 20 }, // 请求参数 dataType: "JSON",
success: function (response, xml) { // 此处放成功后执行的代码 },
fail: function (status) { // 此处放失败后执行的代码 } });
function Ajax(options) {
    options = options || {};
    options.type = (options.type || "GET").toUpperCase();
    options.dataType = options.dataType || "Json";
    var params = formatParams(options.data); // 创建 - 非 IE6 - 第一步

```



```

if (window.XMLHttpRequest) { var xhr = new XMLHttpRequest(); }
else
{ //IE6 及以下版本浏览器 var xhr = new ActiveXObject
('Microsoft.XMLHTTP'); } // 接收 - 第三步
xhr.onreadystatechange = function () {
    if (xhr.readyState == 4) {
var status = xhr.status;
        if (status >= 200 && status < 300) {
options.success && options.success(xhr.responseText, xhr.
responseXML); }
        else { options.fail && options.fail(status);
        }
    }
}
// 连接和发送 - 第二步
if (options.type == "GET") {
    xhr.open("GET", options.url + "?" + params, true); xhr.
send(null); }
else if (options.type == "POST") {
    xhr.open("POST", options.url, true); // 设置表单提交时的内容类型
    xhr.setRequestHeader("Content-Type", "application/x-www-f
orm-urlencoded"); xhr.send(params);
}
}
// 格式化参数
function formatParams(data) {
var arr = []; for (var name in data) {
    arr.push(encodeURIComponent(name) + "=" + encodeURICom
ponent(data[name])); arr.push(("v=" + Math.random()).
replace(".")); return arr.join("&"); }
}

```

40. 写一个函数 padstare(string str1,min_lenthg,string str2), 然后用英文解释每个参数的意思。

答: padstare('5', '3', '0') 返回值是 '005'。

padstare('798', '5', '0') 返回值是 '00798'。

意思是这样的, 如果字符串 str1 的长度没有 min_length 大, 就用 str2 来填充。

41. 写一个命令行字符的解析函数。

答: -name XXX -age XXX -school "XXX" 返回的是 [-name XXX , -age XXX, -school "XXX"]

```

<script>
function getdata(str){
    var Json={};

```

```

        var gets =str.split('')[0];
        gets =gets.split(' ');
        for(var i=0;i<gets.length-1;i++){
            if(i%2==0){
                Json[gets[i]]=gets[i+1];
            }
        }
        Json[gets[gets.length-1]]=str.split('')[1];
        return Json;
    }
</script>

```

42. 设计函数 `indexof(a, b)`，判断字符串 `a` 中是否包含字符串 `b`，如果包含，那么返回其位置；如果不包含，那么返回 `-1`。

答: `<script type="text/javascript">`

```

    function index of(str1,str2){
        var len1=str1.length;
        var len2=str2.length;
        var ret=[];
        if(len1 <len2) {
            return false;
        }else {
            for(var i=0;i<=len1-len2;i++){
                ret.push(str1.substr(i, len2);
                if(ret[i]==str2){return i;}
            }else{return -1;}
        }
    }
</script>

```

43. 有 10 个 ``，在单击任意一个后，打印出单击的 `` 的 `index`。

答: `<script>`

```

    var myul= document.getElementsByTagName("ul")[0];
    var myli= myul.children;
    for(var i=0; i<myli.length; i++){
        myli[i].index = i;
        myli[i].onclick = function(){
            console.log(this.index);
        };
    }
</script>

```

44. 如果是 3.00 元，则转为 300 分；如果是 300 分，则转为 3.00 元。

答：根据传入的数，判断是否为浮点数，是的话返回 `*100`，否的话返回 `/100`，然后 `toFixed`。

```
var num=3.00;// 元
num*=100;// 转为 300 分
num=(num/100).toFixed(2);// 再转为 3.00 元
```

toFixed() 方法把 Number 四舍五入为指定小数位数的数字语法:

```
NumberObject.toFixed(num)
```

45. JS 里面的基础对象和基础数据类有哪些?

答: 数值、字符串型、布尔型、undefined 和空值。

46. 如何在某一个位置插入一个 DIV 对象? 例如: 在下面的 aId 和 bId 间插入一个 DIV 对象。

```
<div>
<div id=" aId" >a</div>
<div id=" bId" >b</div>
<div id=" cId" >c</div>
<div id=" dId" >d</div>
</div>
```

答: window.onload=function(){
var bdiv=document.getElementById('bId');
var box=document.createElement('div');
bdiv.parentNode.insertBefore(box,bdiv);
}

47. 说说 XMLHttpRequest 存有的几个状态 (从 0 到 4 发生的变化)。

答: 0: 请求未初始化;

1: 服务器连接已建立;

2: 请求已接收;

3: 请求处理中;

4: 请求已完成, 且响应已就绪。

48. JavaScript 如何实现多线程计算, 请列举一下常用的方案。

答: 三个常驻线程: JavaScript 引擎线程、界面渲染线程、浏览器事件触发线程。

49. JS 中的三种弹出式消息提醒 (警告窗口、确认窗口、信息输入窗口) 的命令是什么?

答: alert、confirm、prompt。

50. 按照 CommonJS 规范, 在任何模块代码的作用域下设有内置以下哪些变量?

A. module B. context C. require D. exports

答: B

51. 有这样一个 URL: <http://www.qdjh.com/item.htm?a=1&b=2&c=&d=xxx&e>,

请写一个函数并提取 URL 中的各个 GET 参数（参数名和参数个数不确定），将其按 key-value 的形式返回到一个 JSON 结构中，如 {a:'1', b:'2', c:', d:'xxx', e:undefined}。

答：

```
function getUrlParamter() {
    var url = location.search; // 获取 url 中 "?" 符后的字符串
    var obj = new Object();
    if (url.indexOf("?") != -1) {
        var str = url.substr(1);
        strs = str.split("&");
        for(var i = 0; i < strs.length; i ++) {
            obj[strs[i].split("=")[0]]=unescape(strs[i].
split("=")[1]);
        }
    }
    return obj;
}
```

52. 写一个函数，用来判断 1 个整数是否为质数。

解析：

只能被 1 或者本身整除的数叫质数。

```
function iszhishu(num) {
    if (num == 1) {return false;}
    else if (num == 2) { return true;}
    for (var i = 2; i <= Math.sqrt(num); i++) {
        if (num % i == 0) { return false;}
    }
    return true;
}
```

53. 判断字符串是否是这样组成的，第一个必须是字母，后面可以是字母、数字、下划线，总长度为 5 ~ 20。

答：var reg = /^[a-zA-Z][a-zA-Z_0-9]{4,19}\$/;

reg.test("ala__ala__ala__ala__");

54. 截取字符串 abcdefg 中的 efg。

答：var str = "abcdefg";

```
if (/efg/.test(str)) {
    var efg = str.substr(str.indexOf("efg"), 3);
    alert(efg);
}
```

55. 判断一个字符串中出现次数最多的字符，统计其次数。

答：var str = "abcdefgadddda";

```

var obj = {};
for (var i = 0, i = str.length; i <= 1; i++) {
    var key = str[i];
    if (!obj[key]) {
        obj[key] = 1;
    } else {
        obj[key]++;
    }
}
var max = -1;
var max_key = "";
var key;
for (key in obj) {
    if (max < obj[key]) {
        max = obj[key];
        max_key = key;
    }
}
alert("max:"+max+" max_key:"+max_key);

```

56. IE 与 Firefox 脚本有哪些兼容性问题。

答：（1）window.event：表示当前的事件对象，IE 有这个对象，Firefox 没有，Firefox 通过给事件处理函数传递事件对象。

（2）获取事件源：IE 用 srcElement 获取事件源，而 Firefox 用 target 获取事件源。

（3）添加、去除事件：

IE：element.attachEvent(“onclick”，function)

element.detachEvent(“onclick”，function)

Firefox：element.addEventListener(“click”，function, true)

element.removeEventListener(“click”，function, true)

（4）获取标签的自定义属性：

IE：div1.value 或 div1[“value”]

Firefox：可用 div1.getAttribute(“value”)

（5）document.getElementById() 和 document.all[name]

IE：document.getElementById() 和 document.all[name] 均不能获 div 元素，而 Firefox 可以。

（6）input.type 的属性

IE：input.type 只读；

Firefox：input.type 可读写。

（7）是否可用 id 代替 HTML 元素

IE: 可以用 id 来代替 HTML 元素;

Firefox: 不可以用 id 来代替 HTML 元素。

57. 如何避免 JavaScript 多人开发函数重名问题。

答: (1) 分工前应规定命名规范, 然后再根据开发人员开发的功能在函数前加前缀。

(2) 将每个开发人员的函数封装到类中, 需要用的时候直接调用类函数, 只要类名不重复就不会影响整体效果。

58. JavaScript 面向对象中的继承实现一般使用什么?

答: JavaScript 面向对象中的继承实现一般都使用到了构造函数和 Prototype 原型链, 简单的代码如下:

```
function Person(name) {
    this.name = name;
}

function Design(name,book){
    Person.call(this,name);
    This.book=book;
}

Extend(Person,Design)
Design.prototype.getBooks=function(){
    Return this.book;
}
```

59. 在 Firefox 下如何实现 outerHTML ?

答: Firefox 不支持 outerHTML, 要实现 outerHTML 还需要做一些处理。

在页面中添加一个新的元素 A, 复制一份需要获取 outerHTML 的元素, 将这个元素 append 到新的元素中, 然后获取新元素的 innerHTML 就可以了。

60. 编写一个方法, 求一个字符串的字节长度。

答: new function(s) {
 if(!arguments.length||!s) return null;
 if(""==s) return 0;
 var l=0;
 for(var i=0;i <s.length;i++) {
 if(s.charCodeAt(i)>255) l+=2;
 else l++;
 } alert(l);
 }("你");

61. 编写一个方法, 去掉一个数组的重复元素。

答: var arr = [1 ,1 ,2, 3, 3, 2, 1];
 array.prototype.unique = function(){
 var ret = [];

```

var o = {};
var len = this.length;
for (var i=0; i<len; i++){
    var v = this[i];
    if (!o[v]){
        o[v] = 1;
        ret.push(v);
    }
}
return ret;
};
alert(arr.unique());

```

62. 写出 3 个使用 this 的典型应用。

答: (1) 在 HTML 元素事件属性中使用, 如:

```
<input type="button" onclick="showInfo(this);" value=" 单击一下 "/>
```

(2) 构造函数

```

function Animal(name, color) {
    this.name = name;
    this.color = color;
}

```

(3) <input type="button" id="text" value=" 单击一下 " />

```

<script type="text/javascript">
var btn = document.getElementById("text");
btn.onclick = function() {
    alert(this.value);
}
</script>

```

63. DOM 元素如何显示或隐藏?

答: 更改元素的 CSS style, 设为

```

display:'none';
box.style.display = "";
box.style.display = "none";

```

box 是要操作的 DOM 元素。

64. JavaScript 中如何检测一个变量是一个 String 类型? 请写出函数实现。

答: var str = "hello world";

65. 网页中实现一个计算当年还剩多少时间的倒计时程序, 要求网页上实时动态显示 “×× 年还剩 ×× 天 ×× 时 ×× 分 ×× 秒”。

答: <input type="text" value="" id="input" size="1000"/>
 <script type="text/javascript">
 function counter() {

```

var date = new Date();
var year = date.getFullYear();
var date2 = new Date(year, 12, 31, 23, 59, 59);
var time = (date2 - date)/1000;
var day =Math.floor(time/(24*60*60))
var hour = Math.floor(time%(24*60*60)/(60*60))
var minute = Math.floor(time%(24*60*60)%(60*60)/60);
var second = Math.floor(time%(24*60*60)%(60*60)%60);
var str = year + " 年还剩 "+day+" 天 "+hour+" 时
"+minute+" 分 "+second+" 秒 ";
document.getElementById("input").value = str;
}
window.setInterval("counter()", 1000);
</script>

```

66. 补充代码，当鼠标单击 Button1 后，将 Button1 移动到 Button2 的后面

```

<div> <input type="button" id="button1" value="1" onclick="???"> <input type="
button" id="button2" value="2" /> </div>

```

答: <div>

```

<input type="button" id ="button1" value="1" onclick=
"moveBtn(this);">
    <input type="button" id ="button2" value="2" />
</div>
<script type="text/javascript">
function moveBtn(obj) {
    var clone = obj.cloneNode(true);
    var parent = obj.parentNode;
    parent.appendChild(clone);
    parent.removeChild(obj);
}
</script>

```

67. JavaScript 有哪几种数据类型?

答: Number、Boolean、String、Null、Undefined、Object、Array、Function。

68. 下面 CSS 标签在 JavaScript 中调用应如何拼写, border-left-color, -moz-Viewport。

答: borderLeftColor mozViewport

69. JavaScript 中如何对一个对象进行深度 clone?

答: function clone(Obj) {
 var buf;
 if (Obj instanceof Array) {
 buf = []; // 创建一个空的数组


```

        var i = Obj.length;
        while (i--) {
            buf[i] = clone(Obj[i]);
        }
        return buf;
    }else if (Obj instanceof Object){
        buf = {}; // 创建一个空对象
        for (var k in Obj) { // 为这个对象添加新的属性
            buf[k] = clone(Obj[k]);
        }
        return buf;
    }else{
        return Obj;
    }
}

```

70. 如何控制 alert 中的换行。

答: `alert("hello\n world");`

71. 请实现鼠标单击页面中的任意标签, alert 该标签的名称 (注意兼容性)。

答: HTML: `<div id="div">SPANDIV</div>`

`SPAN`

`<p>P</p>`

CSS: `div{ background:#0000FF;width:100px;height:100px;}`

`span{ background:#00FF00;width:100px;height:100px;}`

`p{ background:#FF0000;width:100px;height:100px;}`

JS: `<script type="text/javascript">`

`document.onclick = function(evt){`

`var e = window.event || evt;`

`var tag = e["target"] || e["srcElement"];`

`alert(tag.tagName);`

`};`

`</script>`

72. 请编写一个 JavaScript 函数 `parseQueryString`, 它的用途是把 URL 参数解析为一个对象, 如: `var url="http://www.qdjh.com/index.php?key0=0&key1=1&key2=2";`

答: `function parseQueryString(url){`

`var params = {};`

`var arr = url.split("?");`

`if (arr.length <= 1)`

`return params;`

`arr = arr[1].split("&");`

`for(var i=0, l=arr.length; i<l; i++){`

`var a = arr[i].split("=");`

```

        params[a[0]] = a[1];
    }
    return params;
}
var url = "http://www.qdjhu.com/index.php?key0=0&key1=1&key2=2";
var ps = parseQueryString(url);
alert(ps["key1"]);

```

73. Ajax 是什么？同步和异步的区别？如何解决跨域问题？

答：Ajax 是一种用于创建快速动态网页的技术。通过在后台与服务器进行少量数据交换，可以使网页实现异步更新，这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

同步：脚本会停留并等待服务器发送回复，然后再继续。

异步：脚本允许页面继续其进程，并处理可能的回复。

前端对于跨域的解决办法：

(1) document.domain+iframe。

(2) 动态创建 script 标签。

74. 什么是闭包？下面这个 ul，如何在单击每一列的时候 alert 其 index？

```

<ul id="test">
  <li>这是第一条</li>
  <li>这是第二条</li>
  <li>这是第三条</li>
</ul>

```

答：闭包就是能够读取其他函数内部变量的函数。闭包是可以包含自由（未绑定）变量的代码块，这些变量不是在这个代码块或者任何全局上下文中定义的，而是在定义代码块的环境中定义。

```

(function e() {
    var index = 0;
    var ul = document.getElementById("test");
    var obj = {};
    for (var i = 0, l = ul.childNodes.length; i < l; i++) {
        if (ul.childNodes[i].nodeName.toLowerCase() == "li") {
            var li = ul.childNodes[i];
            li.onclick = function() {
                index++;
                alert(index);
            }
        }
    }
}) ();

```

75. 请给出不少于两种的异步加载 JS 方式。

答:

异步加载方式:

- (1) defer (延迟), 只支持 IE。
- (2) 创建 script, 插入 DOM 中, 加载完毕后回调。

76. 请说出至少三种减少页面加载时间的方法。

答:

- (1) 尽量减少页面中重复的 HTTP 请求数量。
 - (2) 服务器开启 gzip 压缩。
 - (3) CSS 样式的定义放置在文件头部。
 - (4) JavaScript 脚本放在文件末尾。
 - (5) 压缩合并 JavaScript、CSS 代码。
 - (6) 使用多域名负载网页内的多个文件、图片。
77. 请设计一套方案, 用于确保页面中 JS 加载完全。

答:

```
var n = document.createElement("script");
n.type = "text/javascript";
and css nodes)
if(ua.ie){
    n.onreadystatechange = function(){
        var rs = this.readyState;
        if('loaded' === rs || 'complete'===rs){
            n.onreadystatechange = null;
            f(id,url);
        }
    };
n.addEventListener('load',function(){
    f(id,url);
}else{
    n.onload = function(){
        f(id,url);
    };
}
```

78. JS 中如何定义 class, 如何扩展 prototype ?

答:

```
Ele.className = "****"; //*** 在 CSS 中定义 *** {...}
aa.prototype.bb = cc;
```

aa 和 bb 分别是构造函数的名字, 属性; cc 是想要定义的属性的值。

79. 如何添加 HTML 元素的事件, 有几种方法?

答:

- (1) 为 HTML 元素的事件属性赋值。

(2) 在 JS 中使用 `ele.on*** = function() {…}`

(3) 使用 DOM2 的添加事件的方法: `addEventListener` 或 `attachEvent`。

80. 请说出 `document.write` 和 `innerHTML` 的区别。

答: `document.write` 只能重绘整个页面; `innerHTML` 可以重绘页面的一部分。

81. JS 的基础对象有哪些? 将 `window` 和 `document` 常用的方法和属性列出来。

答: 基础对象有: `String`、`Number`、`Boolean`。

`window` 方法:

`setInterval`、`setTimeout`、`clearInterval`、`clearTimeout`。

属性: `name`、`parent`、`screenLeft`、`screenTop`、`self`、`top`、`status`。

`document` 方法:

`createElement`、`getElementById`、`getElementsByName`、`getElementByagName`

属性: `Cookie`、`doctype`、`domain`、`documentElement`、`readyState`、`URL`。

82. 请说出 `Flash`、`Ajax` 各自的优点, 以及在使用中如何取舍?

答:

`Ajax` 的优点:

(1) 可搜索性。

(2) 开放性。

(3) 费用低。

(4) 易用性。

(5) 易于开发。

`Flash` 的优点:

(1) 多媒体处理。

(2) 兼容性。

(3) 矢量图形比 `SVG`、`Canvas` 优势大很多。

(4) 客户端资源调度, 比如麦克风、摄像头。

建议重要和关键部分直接使用 `HTML`, 而交互部分可以使用 `Ajax`。

83. 请使用原生 JS 实现一个 `DIV` 可拖曳, 需要考虑浏览器兼容性。

答: `<div id = "drag1"></div>`

```
<script type="text/javascript">
    window.onload = function() {
        function Drag(obj) {
            this.obj = obj;
        }
        Drag.prototype = {
            constructor: Drag,
            getInitPosition: function(e) {
                e = e || window.event;
                var eX,eY;
```

```

if(e.pageX || e.pageY){
eX = e.pageX;
eY = e.pageY;
}
eX = e.clientX;
eY = e.clientY;
var positionX = eX- this.obj.offsetLeft;
var positionY = eY - this.obj.offsetTop;
return {
x: positionX,
y: positionY
}
},
getmouseCoordinate:function(e) {
e = e || window.event;
if(e.pageX || e.pageY){
return {x:e.pageX, y:e.pageY};
}
return {
x:e.clientX + document.body.scrollLeft - document.body.
clientLeft, y:e.clientY + document.body.scrollTop - document.
body.clientTop
};
},
initDrag:function() {
var tempThis = this;
this.obj.onmousedown = function(e) {
var initP = tempThis.getInitPosition();
document.onmousemove = function(e) {
var moveP = tempThis.getmouseCoordinate();
tempThis.obj.style.marginTop = moveP.y - initP.y + "px";
tempThis.obj.style.marginLeft = moveP.x - initP.x + "px";
}
document.onmouseup = function(){
document.onmousemove = null;
document.onmouseup = null;
} }
} }
var drag = document.getElementById("drag1");
var dragElement = new Drag(drag);
dragElement.initDrag();
}
</script>

```

84. 下面哪些是 NodeJS 官方模块 () (多选)

A. Querystring B. Dns C. Async D. Request

答: AB

85. 如何通过 Ajax 来判断浏览器是 IE 浏览器还是火狐浏览器, 用 Ajax 实现。

答: 要想通过 Ajax 来判断是 IE 浏览器还是 Firefox 浏览器, 就应该通过 XMLHttpRequest 对象。

```
var xmlhttp;if (window.XMLHttpRequest) {
    xmlhttp = new XMLHttpRequest();
    alert("your brower is not IE ");} else {
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    alert("your brower is IE ")}
```

jQuery 题型

86. 怎样实现前端优化?

答:

- (1) 减少 http 请求。
- (2) 使用内容发布网络。
- (3) 添加 expires 头。
- (4) 压缩组件。
- (5) 样式表放在头部, 脚本放在底部, 避免 CSS 表达式——精简 JS/CSS。
- (6) 减少 DNS 查找。
- (7) 避免重定向。
- (8) 配置或者移除 Etag。
- (9) 使用 Ajax 可缓存。

87. 你知道 DOM 的年份吗? DOM 有什么优点和缺点?

答: 在 1998 年, W3C 发布了第一级的 DOM 规范。这个规范允许访问和操作 HTML 页面中的每一个单独的元素。

W3C DOM 标准被分为 3 个不同的部分:

- 核心 DOM——针对任何结构化文档的标准模型
- XML DOM——针对 XML 文档的标准模型
- HTML DOM——针对 HTML 文档的标准模型

DOM 的优点主要表现在: 易用性强, 使用 DOM 时, 能把所有的 XML 文档信息都存储于内存中; 增强了易用性; 遍历简单, 支持 XPath。

DOM 的缺点主要表现在: 效率低、解析速度慢, 内存占用量过高, 对于大文件来说几乎不可能使用; 消耗时间, 因为使用 DOM 进行解析时, 将为文档的每个 element、attribute、processing-instrUction 和 comment 都创建一个对象, 这

样在 DOM 机制中所运用的大量对象的创建和销毁无疑会消耗大量的时间。

88. 列举常用的浏览器类型，以及它们使用的内核，还有对应的调试工具。

答：常用的浏览器有 IE(6,7,8,9,10),Firefox,chrome。

三者内核各不相同。

IE 基于 IE 内核，常用的调试工具有 IEWebDeveloper（IE9 默认有安装）。

Firefox 大家估计用得最多。Firefox 基于 Mozilla 内核，专业的前端开发人员一般都会安装一个 Firebug 插件来进行调试。

Chrome（内核 webkit）自带的有 google 开发的内置调试工具。

其他还有 Opera、遨游、世界之窗等。

Chrome 内核跑得比较快，且更安全。

Firefox 做调试是最棒的。

89. 解释一下什么是 Web 语义化，举出具体的实例，并说明语义化后有什么好处？

答：Web 里的概念，如：

<h1 2 3 4 5> 这里常用的是标题，就不要用 DIV 标签了；

或 li 列表，就不要用 <p> 或 <td> 了。

好处：易理解、渲染快、利于 seo 优化。

90. 说出 IE 浏览器和其他浏览器在页面元素引用有什么区别？

答：这个与内核及是否由 W3C 来定制有关系，不同浏览器渲染结果不同。

目前国内还有大部分用户在使用 IE6，所以 Web 在制作的时候常常会碰到兼容性的问题，如：display-block、padding、margin 等盒子模型比较多，还有不同的字间距等产生的问题。

常用解决的方法：IE6：_xxx:{} IE7：* IE 和其他不同浏览器间的差异。

91. 为什么利用多个域名来存储网站资源会更有效？

答：确保用户在不同地区能用最快的速度打开网站，即使其中某个域名崩溃，用户也能通过其他域名访问网站。

92. 浏览器一次可以从一个域名下做多少资源？

答：只能够读取到这个服务器相关的信息，而且浏览器一般只允许存放 300 个 Cookie，每个站点最多存放 20 个。现在每个 Cookie 的大小为 4KB，根本不会占用多少空间。

93. 你知道哪些降低页面加载时间的方法？

答：（1）压缩 CSS、JS 文件；

（2）合并 JS、CSS 文件，减少 http 请求；

（3）外部 JS、CSS 文件放在底下。

94. 如果你接到了一个使用 Tab 来缩进代码的项目，但是你喜欢使用空格，你会怎么做？

答：为了保持一致性，转换成项目原有的风格。

95. 今年你打算熟练掌握一项新技术，你会学习哪门技术？

答：NodeJS。现如今在 IT 行业上 JavaScript 逐渐走向前端，NodeJS 是一项非常重要的创新，服务器端也可以使用 NodeJS。

96. 说一说你对网页标准和标准制定机构重要性的理解。

答：网页标准和标准制定机构都是为了能让 Web 发展得更“健康”，开发者遵循统一的标准，降低开发难度，减少开发成本，SEO 也会更好做，并且不会因为滥用代码导致各种 Bug 或产生各种安全问题，最终提高网站易用性。

97. 什么是 FOUC？如何避免 FOUC？

答：FOUC（Flash Of Unstyled Content，文档样式闪烁），IE 在加载网页时，CSS 样式会短暂失效。

在 head 里面添加一个 link 或者 script 标签，如下：

```
<link rel="stylesheet" type="text/css" media="print" href="print.css">  
  
<script type="text/javascript"> </script>
```

98. 前端开发的优化问题。

答：

- (1) 减少 http 请求次数。
- (2) JS、CSS 源码压缩。
- (3) 前端用变量保存 Ajax 请求结果，每次操作本地变量，不用请求。
- (4) 用 innerHTML 代替 DOM 操作，减少 DOM 操作次数，优化 JavaScript 性能。
- (5) 用 setTimeout 来避免页面失去响应。
- (6) 当需要设置的样式很多时，设置 className，而不是直接操作 style。
- (7) 尽可能少地使用全局变量。
- (8) 缓存 DOM 节点查找的结果。
- (9) 避免使用 CSS Expression。
- (10) 避免在页面的主体布局中使用 table。table 要等其中的内容完全下载之后才会显示出来，显示比 DIV+CSS 布局慢。

99. 如何控制网页在网络传输过程中的数据量？

答：最显著的方法是启用 Gzip 压缩。此外养成编码的好习惯，避免重复的代码、HTML 标签和属性。

100. 多浏览器通过什么进行检测？

答：需要设置一下网络连接的环境。

101. 说一说 HTML5 发展的前景和会遇见的瓶颈。

答：前景：HTML5 是大趋势，响应式布局，NodeJS 高并发后台处理。

瓶颈：HTML5 运行速度较慢，对浏览器要求较高，浏览器兼容性问题。

102. HTML5 和 CSS3 有什么新特性？

答：HTML5 强化了 Web 网页的表现性能，如 :nav、header、section、canvas 等，语义化更强。

CSS3 新特性有阴影特效、圆角处理等，都是非常不错的效果。

103. table 标签中的 border、cellpadding 和 td 标签中的 colspan、rowspan 分别起什么作用？

答：border——边界

cellpadding——边距

colspan——跨列数

rowspan——跨行数

104. 你做的页面在哪些浏览器测试过？这些浏览器的内核分别是什么？

答：IE6、IE7、IE8、Firefox、Opera、Safari、Chrome、Maxthon。

Trident：Windows 下的 IE 浏览器使用的内核代号。除 IE 外，众多的 IE Shell（如 Maxthon）都使用这个内核。

Gecko：Mozilla Firefox 浏览器使用的内核代号。

Presto：Opera 浏览器使用的内核代号，这是目前公认网页浏览速度最快的浏览器内核。

KHTML/WebCore：Konqueror/Safari 浏览器使用的内核代号。

Webkit：Chrome 浏览器使用的内核代号。Safari 也可使用。

Maxthon 浏览器是基于 IE 内核的。

105. 在每个 HTML 文件里，开头都有个很重要的东西——<!DOCTYPE>，知道这是干什么的吗？

答：<!DOCTYPE> 声明位于文档中的最前面的位置，处于 <html> 标签之前此标签可告知浏览器文档使用哪种 HTML 或 XHTML 规范。

106. Quirks 模式是什么？它和 Standards 模式有什么区别？

答：从 IE6 开始，引入了 Standards 模式，标准模式中，浏览器尝试给符合标准的文档在规范上的正确处理达到在指定浏览器中的程度。

在 IE6 之前，CSS 还不够成熟，所以在 IE5 之前的浏览器对 CSS 的支持很差，IE6 能对 CSS 提供更好的支持，然而此时问题就来了，因为有很多页面是基于旧的布局方式写的，而如果 IE6 支持 CSS，则将令这些页面不能正常显示，如何既保证不破坏现有页面，又能提供新的渲染机制呢？

在写程序时，我们也会经常遇到这样的问题，如何保证原来的接口不变，又提供更强大的功能，尤其是当新功能不兼容旧功能时。遇到这种问题时说道，常见的做法是增加参数和分支，即当某个参数为真时，我们就使用新功能，而如果这个参数不为真时，就使用旧功能，这样就不仅不会破坏原有的程序，

而且提供了新功能。IE6 也是类似这样做的，它将 DTD 当成了这个“参数”，因为在以前的页面，大家都不会去写 DTD，所以 IE6 假定：如果写了 DTD，就意味着这个页面将采用对 CSS 支持更好的布局，而如果没有，则采用兼容之前的布局方式。这就是 Quirks 模式（怪癖模式、诡异模式、怪异模式）。

区别：总体会有布局、样式解析和脚本执行三个方面的区别。

107. 相对 TABLE 布局，DIV+CSS 布局有哪些优点？

答：（1）改版的时候更方便，只要改 CSS 文件即可。

（2）页面加载速度更快、结构化清晰、页面显示简洁。

（3）表现与结构相分离。

（4）易于优化，搜索引擎更友好，排名更容易靠前。

108. img 的 alt 与 title 有何异同？strong 与 em 有何异同？

答：alt 与 title 的异同：

alt(alt text): 为不能显示图像、窗体或 applets 的用户代理 (UA)，alt 属性用来指定替换文字。替换文字的语言由 lang 属性指定。（在 IE 浏览器下，会在没有 title 时把 alt 当成 tool tip 显示）

title(tool tip): 该属性为设置该属性的元素提供建议性的信息。

Strong 与 em 的异同：

strong: 粗体强调标签，表示内容的重要性。

em: 斜体强调标签，强调更强烈，表示内容的强调点

109. 你知道渐进增长和优雅降级的不同吗？

答：渐进增强 (progressive enhancement)：

针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能，达到更好的用户体验。

优雅降级 (graceful degradation)：

一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。

区别：优雅降级是从复杂的现状开始，并试图减少用户体验的供给，而渐进增强则是从一个非常基础的、能够起作用的版本开始，并不断扩充，以适应未来环境的需要。降级（功能衰减）意味着往回看；而渐进增强则意味着朝前看，同时保证其根基处于安全地带。

110. 为什么要通过多个域名来存储网站资源？

答：（1）CDN 缓存更方便；

（2）突破浏览器并发限制；

（3）节约 Cookie 带宽；

（4）节约主域名的连接数，优化页面响应速度；

（5）防止不必要的安全问题。

111. 对网页标准和标准制定机构，你有什么看法？

答：网页标准和标准制定机构都是为了让 Web 发展得更“健康”，开发者遵循统一的标准，降低开发难度和开发成本，此时 SEO 也会更好做，并且不会因为滥用代码导致各种 Bug、安全问题，最终提高网站易用性。（仅供参考，无标准答案）

112. sessionStorage 和 localStorage、Web Storage 和 Cookie，之间有什么区别？

答：sessionStorage 用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访问，并且当会话结束后数据也随之销毁。因此 sessionStorage 不是一种持久化的本地存储，仅仅是会话级别的存储。而 localStorage 用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

Web Storage 的概念和 Cookie 相似，区别是，它是为了更大容量存储而设计的。Cookie 的大小是受限的，并且当每次请求一个新的页面时，Cookie 都会被发送过去，这样无形中浪费了带宽，另外 Cookie 还需要指定作用域，不可以跨域调用。

除此之外，Web Storage 拥有 setItem、getItem、removeItem、clear 等方法，不像 Cookie 需要前端开发者自己封装 setCookie、getCookie。但是 Cookie 是不可或缺的。Cookie 的作用是与服务器进行交互，作为 HTTP 规范的一部分而存在，而 Web Storage 仅仅是为了在本地“存储”数据而生。

113. src 与 href 都是链接方式，它们有什么不同之处？

答：src 用于替换当前元素，href 用于在当前文档和引用资源之间确立联系。

src 是 source 的缩写，指向外部资源的位置，指向的内容将会嵌入到文档中当前标签所在位置；在请求 src 资源时会将其指向的资源下载并应用到文档内，例如 js 脚本、img 图片和 frame 等元素。

```
<script src="js.js"></script>
```

当浏览器解析到该元素时，会暂停其他资源的下载和处理，直到将该资源加载、编译、执行完毕，图片和框架等元素也如此，类似于将所指向资源嵌入当前标签内。这也是为什么将 js 脚本放在底部而不是头部。

href 是 Hypertext Reference 的缩写，指向网络资源所在位置，建立和当前元素（锚点）或当前文档（链接）之间的链接，如果我们在文档中添加

```
<link href="common.css" rel="stylesheet"/>
```

那么浏览器会识别该文档为 CSS 文件，就会并行下载资源，并且不会停止对当前文档的处理。这也是为什么建议使用 link 方式来加载 CSS，而不是使用 @import 方式。

114. 网页制作支持哪些格式的图片？

答：png-8、png-24、jpeg、gif、svg。

但是上面的那些都不是面试官想要的最好的答案。面试官希望听到是 Webp、Apng。（是否有关新技术、新鲜事物）

科普一下，Webp：WebP 格式，谷歌（Google）开发的一种旨在加快图片加载速度的图片格式。图片压缩体积大约只有 jpeg 的 2/3，并能节省大量的服务器带宽资源和数据空间。Facebook eBay 等知名网站已经开始测试并使用 WebP 格式。

在质量相同的情况下，WebP 格式图像的体积要比 jpeg 格式图像小 40%。

Apng：全称是“Animated Portable Network Graphics”，是 png 的位图动画扩展，可以实现 png 格式的动态图片效果。2004 年诞生，但一直得不到各大浏览器厂商的支持，如今得到了 iOS safari 8 的支持，有望代替 GIF 成为下一代动态图标准。

115. 请谈谈你对微格式的理解。

答：微格式（Microformats）是一种让机器可读的语义化 XHTML 词汇的集合，是结构化数据的开放标准。它是为特殊应用而制定的特殊格式。

优点：将智能数据添加到网页上，让网站内容在搜索引擎结果界面可以显示额外的提示。（应用范例：豆瓣，有兴趣的读者可以自行在网上搜索）

116. 在 CSS/JS 代码上线之后，开发人员经常会优化性能，从用户刷新网页开始，请求一次 JS，会有哪些地方出现缓存处理？

答：dns 缓存、cdn 缓存、浏览器缓存、服务器缓存。

117. 当一个大型网站上有大量的图片时，要使这些图片加载速度变快，以达到更好的体验，有哪些方法？

答：图片懒加载，即在页面上的未可视区域添加一个滚动条事件，判断图片位置与浏览器顶端的距离，以及图片位置与页面的距离，如果前者小于后者，则优先加载。

如果为幻灯片、相册等，可以使用图片预加载技术，将当前展示图片的前一张和后一张优先下载。如果图片为 CSS 图片，可以使用 CSS Sprite、SVG Sprite、Icon Font、Base64 等技术。如果图片过大，可以使用特殊编码的图片，加载时会先加载一张压缩得特别厉害的缩略图，以提高用户体验。如果图片展示区域小于图片的真实大小，则在服务器端根据业务需要先行进行图片压缩，图片压缩后大小与展示一致。

118. 对 HTML 的结构语义化，你是怎样理解的？

答：去掉或样式丢失时，能让页面呈现清晰的结构。

HTML 本身是没有表现的，例如 <h1> 是粗体，字体大小为 2em，加粗； 是加粗的，不要认为这是 HTML 的表现，其实是 HTML 默认的 CSS 样式在起作用，所以去掉或样式丢失时能让页面呈现清晰的结构不是语义化的 HTML 结构的优点，但是浏览器都有默认样式，默认样式的目的也是为了更好

地表达 HTML 的语义，可以说浏览器的默认样式和语义化的 HTML 结构是不可分割的。

119. 想要做好 SEO，站在前端开发人员的角度需要考虑什么？

答：

- (1) 了解搜索引擎如何抓取网页和如何索引网页。
- (2) Meta 标签优化。
- (3) 如何选取关键词，并在网页中放置关键词。
- (4) 了解主要的搜索引擎。
- (5) 主要的互联网目录。
- (6) 按单击付费的搜索引擎。
- (7) 搜索引擎登录。
- (8) 链接交换和链接广泛度 (Link Popularity)。
- (9) 合理的标签使用。

120. 对 DOM 设置 CSS 样式时，有哪些方式？

答：

- (1) 外部样式表，引入一个外部 CSS 文件。
- (2) 内部样式表，将 CSS 代码放在 <head> 标签内部。
- (3) 内联式，将 CSS 样式直接定义在 HTML 元素内部。

121. 请列举 CSS 中的选择器。

答：(1) 派生选择器（用 HTML 标签申明）；

(2) ID 选择器（用 DOM 的 ID 申明）；

(3) 类选择器（用一个样式类名申明）；

(4) 属性选择器（用 DOM 的属性申明，属于 CSS2，IE6 不支持，不常用）；

(5) 后代选择器（利用空格间隔，比如 `div .a{ }`）；

(6) 群组选择器（利用逗号间隔，比如 `p,div,#a{ }`）。

122. 在 CSS 中对于选择器的优先级是怎样定义的？

答：一般而言，选择器越特殊，它的优先级越高。也就是说，选择器指向越准确，它的优先级就越高。

用 1 表示派生选择器的优先级；

用 10 表示类选择器的优先级；

用 100 表示 ID 选择器的优先级。

`div.test1 .span var` 优先级 $1+10+10+1$ 。

`span#xxx .songs li` 优先级 $1+100+10+1$ 。

`#xxx li` 优先级 $100+1$ 。

123. 请看以下代码，并说出这时 p 标签内的文字是什么颜色？

```
<style>
  .colorA{ color:blue;}
  .colorB{ color:red;}
</style>
<body>
  <p class='colorB colorA'> 123 </p>
</body>
```

答：红色

124. CSS 中有哪些属性可以让 DOM 元素在浏览器中消失？

答：（1）最基本的：设置 display 属性为 none，或者设置 visibility 属性为 hidden。

（2）技巧性的：设置宽和高为 0，设置透明度为 0，设置 z-index 位置在 -1000。

125. hover 样式在超链接访问过后为什么就没有了，请说出解决方法。

答：被单击访问过的超链接样式不再具有 hover 和 active 了，解决方法是改变 CSS 属性的排列顺序：L-V-H-A（link-visited-hover-active）。

126. CSS 中的 Hack 指的是什么？

答：针对不同的浏览器写不同的 CSS code 的过程，就是 CSS Hack。

127. CSS3 中的哪个属性可以做一个简单的幻灯片效果。

答：使用 animation 动画。

128. 块级元素和内联元素之间的特性是什么？内联元素的 padding 和 margin 可以设置吗？

答：块级元素（block）特性：

（1）总是独占一行，表现为另起一行开始，而且其后的元素也必须另起一行显示。

（2）宽度（width）、高度（height）、内边距（padding）和外边距（margin）都可控制。

内联元素（inline）特性：

（1）和相邻的内联元素在同一行。

（2）宽度（width）、高度（height）、内边距的宽度（width）、高度（height）、内边距（padding）和外边距（margin）都可控制；top/bottom(padding-top/padding-bottom) 和外边距的 top/bottom(margin-top/margin-bottom) 都不可改变，就是里面文字或图片的大小。

padding 和 margin 的 left 和 right 是可以设置的。

129. 在 CSS 中什么是外边距重叠？重叠之后的效果又是什么？

答：外边距重叠就是 margin-collapse。

在 CSS 中，相邻的两个盒子（可能是兄弟关系，也可能是祖先关系）的外

边距可以重叠成一个单独的外边距，这种合并外边距的方式被称为折叠，并且因而所结合成的外边距称为折叠外边距。

折叠结果遵循下列计算规则：

1. 两个相邻的外边距都是正数时，折叠结果是它们两者之间较大的值。
2. 两个相邻的外边距都是负数时，折叠结果是两者绝对值的较大值。
3. 两个外边距一正一负时，折叠结果是两者相加的和。

130. `rgba()` 与 `opacity()` 都是能够实现透明效果的，请问在实现透明效果时有什么区别？

答：`rgba()` 和 `opacity()` 都能实现透明效果，但最大的不同是 `opacity()` 作用于元素，以及元素内的所有内容的透明度；而 `rgba()` 只作用于元素的颜色或其背景色。

131. 在 CSS 中能够让文字垂直和水平方向重叠的两个属性分别是什么？

答：（1）垂直方向：`line-height`；

（2）水平方向：`letter-spacing`。

132. 请用代码实现一个 ID 名为 `div_1` 的浮动元素是怎样垂直居中的？

答：`#div_1{width: 200px; height: 200px; background-color: #6699FF;`

`margin:auto; position: absolute; left: 0; top: 0; right: 0;`

`bottom: 0; }`

133. 请用代码实现一个让 ID 名为 `content` 的元素 `tgf` 垂直居中。

答：`#container #content { display: table-cell; text-align: center; vertical-align: middle; }`

134. 请问 `px` 和 `em` 有什么异同？

答：`px` 和 `em` 都是长度单位，区别是，`px` 的值是固定的，指定多少就是多少，计算比较容易。而 `em` 的值不是固定的，并且 `em` 会继承父级元素的字体大小。

浏览器的默认字体高都是 16px，所以未经调整的浏览器都符合：`1em=16px`。那么 `12px=0.75em`，`10px=0.625em`。

135. 请具体说一下什么是“reset”（指 CSS 文件），并解释一下“reset”是如何使用的，同时请与 `Normalize.css` 进行对比，了解它俩之间的不同之处。

答：“reset”是重置样式，重置样式非常多，凡是一个前端开发人员肯定有一个常用的重置 CSS 文件，并知道如何使用它们。他们是盲目地使用还是有目的地使用呢？由于不同的浏览器对一些元素有不同的默认样式，如果你不处理，在不同的浏览器下会存在必要的风险，所以我们要有目的地使用 `reset`。

我们可以用 `Normalize` 来代替重置样式文件。它没有重置所有的样式风格，仅提供了一套合理的默认样式值。既能让众多浏览器达到一致和合理，又不扰

乱其他的東西。

136. 請描述一下 Sass、LESS 是什麼，為什麼會頻繁地使用它們？

答：它們是 CSS 預處理器，是 CSS 上的一種抽象層。它們把一種特殊的語法 / 語言編譯成 CSS。

優點：

(1) 結構清晰，便於擴展。

(2) 可以方便地屏蔽瀏覽器私有語法差異。封裝對瀏覽器語法差異的重複處理，減少無意義的機械勞動。

(3) 可以輕鬆實現多重繼承。完全兼容 CSS 代碼，可以方便地應用到老項目中。LESS 只是在 CSS 語法上做了擴展，所以老的 CSS 代碼也可以與 LESS 代碼一同編譯。

137. 請分別列舉 display: none 與 visibility: hidden 的原理（也就是所謂的區別）。

答：display：隱藏對應的元素，但不擠占該元素原來的空間。

visibility：隱藏對應的元素，並且擠占該元素原來的空間。

使用 CSS display:none 屬性後，HTML 元素（對象）的寬度、高度等各種屬性值都將“丟失”；而使用 visibility:hidden 屬性後，HTML 元素（對象）僅僅是在視覺上看不到（完全透明），但是它所佔據的空間位置仍然存在。

138. 在 CSS 中的 content 屬性有什麼作用？。

答：CSS 的 content 屬性專門應用在 before/after 偽元素上，用於插入生成內容。最常見的應用是利用偽類清除浮動。

after 偽元素通過 content 在元素的後面生成了內容為一個點的塊級元素，再利用 clear:both 清除浮動。

139. 請使用 CSS content 屬性來實現 CSS 計數器。

答：CSS 計數器是通過設置 counter-reset、counter-increment 兩個屬性，以及 counter()/counters() 的方法，配合 after / before 偽類來實現的。

140. iframe 有哪些缺點？

(1) iframe 會阻塞主頁面的 Onload 事件。

(2) iframe 和主頁面共享連接池，而瀏覽器對相同域的連接有限制，所以會影響頁面的並行加載。

使用 iframe 之前需要考慮這兩個缺點。如果需要使用 iframe，最好是通过 JavaScript

動態給 iframe 添加 src 屬性值，這樣可以繞開以上兩個問題。

141. 使用 CSS 來實現一個頭部和尾部是固定的而中間部分是自適應效果的佈局。

答：<div class="content">


```

<div class="header"></div>
    <div class="main"></div>
    <div class="foot"></div>
</div>
<style>
    *{margin:0; padding:0;}
    html,body,.con{height:100%;width:100%;height:100%;width:100%;}
    div{position:absolute;}
    .header,.foot{width:100%;height:100px;z-index:5;}
    .header{background:red;top:0;}
    .bottom{background:black;bottom:0;}
    .main{ width:100%; background:#a7fad7;overflow:auto;top:100px;
    bottom:100px;position:absolute;_height:100%;_border-top:-100px solid #eee; _border-bottom:-100px solid #eee; _top:0; }
</style>

```

jQuery 题型

142. jQuery 中的美元符号 \$ 起什么作用?

答：其实美元符号 \$ 只是“jQuery”的别名，它是 jQuery 的选择器，如下面的代码：

```
$(document).ready(function(){});
```

当然，你也可以用 jQuery 来代替 \$，如下面的代码：

```
jQuery(document).ready(function(){});
```

143. onload() 函数与 document.ready() 函数之间的区别。

答：（1）我们可以在页面中使用多个 document.ready()，但只能使用一个 onload()。

（2）document.ready() 函数在页面 DOM 元素加载完以后就会被调用，而 onload() 函数则要在所有的关联资源（包括图像、音频）加载完毕后才会调用。

144. 在 jQuery 中一共有几种类型的选择器，请一一列举出来。

答：（1）基本过滤选择器；

（2）属性过滤选择器；

（3）内容过滤选择器；

（4）子元素过滤选择器；

（5）可见性过滤选择器；

（6）表单元素过滤选择器。

145. 将元素边框设置为 2px 宽的虚线（使用 jQuery）。

答：<script language="javascript" type="text/javascript">

```

    $("*").css("border", "2px dotted red");
</script>

```

146. 说一说当 jQuery 的文件不能在 CDN 上使用的一些操作。

答：为了节省带宽和脚本引用的稳定性，我们会使用 CDN 上的 jQuery 文件，例如 Google 的 jQuery CDN 服务。但是如果这些 CDN 上的 jQuery 服务不可用，我们还可以通过以下代码来切换到本地服务器的 jQuery 版本：

```

<script type="text/JavaScript" language="JavaScript"
    src="http://www.qdjh.com/Ajax/jquery/jquery-1.4.1.min.js"
"></script>
    <script type='text/javascript'>//
    if (typeof jQuery == 'undefined') {
    document.write(unescape("%3Cscript src='/Script/jquery-
    1.4.1.min.js'
        type='text/javascript' %3E%3C/script%3E"));
    }
    ]]&gt;
&lt;/script&gt;
</pre>
</div>
<div data-bbox="172 433 814 450" data-label="Text">
<p>147. 在 jQuery 中，如何实现单击一个按钮可以出现一个对话框效果。</p>
</div>
<div data-bbox="171 455 641 471" data-label="Text">
<p>答：&lt;input id="file" type="text" size="12"/&gt;</p>
</div>
<div data-bbox="213 478 576 623" data-label="Text">
<pre>
jQuery:
&lt;script type="text/javascript"&gt;
    $(document).ready(function () {
        $('#Btn1').click(function () {
            alert($('#file').attr("value"));
        });
    });
&lt;/script&gt;
</pre>
</div>
<div data-bbox="172 627 673 644" data-label="Text">
<p>148. delegate() 函数和在 jQuery 中起着什么样的作用？</p>
</div>
<div data-bbox="128 648 870 688" data-label="Text">
<p>答：如果你有一个父元素，需要给其下的子元素添加事件，这时你可以使用 delegate()，代码如下：</p>
</div>
<div data-bbox="213 694 684 727" data-label="Text">
<pre>
$("ul").delegate("li", "click", function() {
    $(this).hide();
</pre>
</div>
<div data-bbox="172 731 761 748" data-label="Text">
<p>149. 在 jQuery 中，使用什么方法来实现对 URL 的解码和编码。</p>
</div>
<div data-bbox="172 753 813 770" data-label="Text">
<p>答：在 jQuery 中，可以使用以下方法来实现对 URL 进行编码和解码：</p>
</div>
<div data-bbox="213 776 770 791" data-label="Text">
<pre>
encodeURIComponent(url) and decodeURIComponent(url)
</pre>
</div>
<div data-bbox="172 795 660 811" data-label="Text">
<p>150. 请写出一段代码来禁用浏览器前进后退的按钮。</p>
</div>
<div data-bbox="171 816 794 891" data-label="Text">
<p>答：&lt;script type="text/javascript" language="javascript"&gt;</p>
<pre>
    $(document).ready(function() {
        window.history.forward(1);
        window.history.forward(-1);
</pre>
</div>
<div data-bbox="128 920 173 935" data-label="Page-Footer">
<p>348</p>
</div>
```

```
});  
</script>
```

二、效果题

151. 什么叫优雅降级和渐进增强？解释：

渐进增强（progressive enhancement）：

针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能以达到更好的用户体验。

优雅降级（graceful degradation）：

一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。

区别：

- （1）优雅降级是从复杂的现状开始的，并试图减少用户体验的供给。
- （2）渐进增强则是从一个非常基础的、能够起作用的版本开始，并不断扩充，以适应未来环境的需要。
- （3）降级（功能衰减）意味着往回看；而渐进增强则意味着朝前看，同时保证其根基处于安全地带

152. text-shadow 属性的 4 个值分别是什么？

答：

语法：

text-shadow: h-shadow v-shadow blur color;

注释：text-shadow 属性向文本添加一个或多个阴影。该属性是逗号分隔的阴影列表，每个阴影由两个或三个长度值和一个可选的颜色值进行规定。省略的长度是 0。

- （1）h-shadow：必需。水平阴影的位置，允许负值。
- （2）v-shadow：必需。垂直阴影的位置，允许负值。
- （3）blur：可选。模糊的距离。
- （4）color：可选。阴影的颜色，参阅 CSS 颜色值。

153. '5'+4=？

答：54。

154. 如何用 JS 判断浏览器类型？

答：现在网络上的浏览器和操作系统就像中国的方言一样，非常之多，这给开发人员带来了巨大的痛苦。大家的喜好不同，用的系统也不同！有人喜欢用 IE，有人喜欢用 Firefox，还有人喜欢用腾讯 TT，有的人喜欢用 maxthon。虽然名字有很多种，但是内核却只有几种，如：IE 内核、netscape 内核等。那么怎么样用 JS 来判断各种浏览器的类型呢？请见如下代码。

```
<script language="JavaScript">
```

```
function getOs()
{
    var OsObject = "";
    if(navigator.userAgent.indexOf("MSIE")>0) {
        return "MSIE";
    }
    if(isFirefox=navigator.userAgent.indexOf("Firefox")>0){
        return "Firefox";
    }
    if(isSafari=navigator.userAgent.indexOf("Safari")>0) {
        return "Safari";
    }
    if(isCamino=navigator.userAgent.indexOf("Camino")>0){
        return "Camino";
    }
    if(isMozilla=navigator.userAgent.indexOf("Gecko/")>0){
        return "Gecko";
    }
}
alert(" 您的浏览器类型为:"+getOs());
</script>
```

155. 用 JS 写一个当年还剩多少时间的倒计时程序。

答: <input type=" text" value="" id=" input" size=" 1000" />
 <script type=" text/javascript" >
 function counter() {
 var date = new Date();
 var year = date.getFullYear();
 var date2 = new Date(year, 12, 31, 23, 59, 59);
 var time = (date2 - date)/1000;
 var day =Math.floor(time/(24*60*60))
 var hour = Math.floor(time%(24*60*60)/(60*60))
 var minute = Math.floor(time%(24*60*60)%(60*60)/60);
 var second = Math.floor(time%(24*60*60)%(60*60)%60);
 var str = year + " 年还剩 "+day+" 天 "+hour+" 时 "+minute+" 分 "+second+" 秒";
 document.getElementById("input").value = str;
 }
 window.setInterval("counter()", 1000);
 </script>
 </script>

18.2 前端面试模拟试题二

General Questions:

第 1 道: Are you on Twitter? If so, who do you follow on Twitter?

第 2 道: Are you on Github? If so, what are some examples of repos you follow

第 3 道: What blogs do you follow?

第 4 道: What version control systems have you used?

第 5 道: What is your preferred development environment? (OS, Editor, Browsers, Tools etc.)

第 6 道: Can you describe your workflow when you create a web page?

第 7 道: Can you describe the difference between progressive enhancement and graceful degradation?

Bonus points for the answer "no one can".

Extra bonus points for describing feature detection.

第 8 道: Explain what "Semantic html" means.

第 9 道: What does "minification" do?

第 10 道: Why is it better to serve site assets from multiple domains?

How many resources will a browser download from a given domain at a time?

第 11 道: If you have 8 different stylesheets for a given design, how would you integrate them into the site?

Looking for file concatenation.

Points off for @import, unless it works in conjunction with a build system.

第 12 道: If you jumped on a project and they used tabs and you used spaces, what would you do?

issue :retab! command.

第 13 道: Write a simple slideshow page.

Bonus points if it does not use JS.

第 14 道: What tools do you use to test your code's performance?

第 15 道: If you could master one technology this year, what would it be?

第 16 道: Name 3 ways to decrease page load. (perceived or actual load time)

第 17 道: Explain the importance of standards.

HTML-Specific Questions:

第 18 道: What's a doctype do, and how many can you name?

第 19 道: What's the difference between standards mode and quirks mode?

第 20 道: What are the limitations when serving XHTML pages?

第 21 道：How do you serve a page with content in multiple languages?

第 22 道：Can you use XHTML syntax in HTML5? How do you use XML in HTML5?

第 23 道：What are data-attributes good for?

第 24 道：What are the content models in HTML4 and are they different in HTML5?

第 25 道：Consider HTML5 as an open web platform. What are the building blocks of HTML5?

第 26 道：Describe the difference between Cookies, sessionStorage and localStorage.

JS-Specific Questions:

第 27 道：Which JavaScript libraries have you used?

第 28 道：How is JavaScript different from Java?

第 29 道：What are undefined and undeclared variables?

第 30 道：What is a closure, and how/why would you use one?

Your favorite pattern used to create them? argyle (Only applicable to IIFEs).

第 31 道：What's a typical use case for anonymous functions?

第 32 道：Explain the "JavaScript module pattern" and when you'd use it.

Bonus points for mentioning clean namespacing.

What if your modules are namespace-less?

第 33 道：How do you organize your code? (module pattern, classical inheritance?)

第 34 道：What's the difference between host objects and native objects?

第 35 道：What's the difference between .call and .apply?

第 36 道：Explain Function.prototype.bind?

第 37 道：When do you optimize your code?

第 38 道：Can you explain how inheritance works in JavaScript?

Bonus points for the funny answer: "no one can".

Extra bonus points if they take a stab at explaining it.

第 39 道：When would you use document.write()?

Correct answer: 1999 – time to weed out the junior devs.

第 40 道：What's the difference between feature detection, feature inference, and using the UA string.

第 41 道：Explain Ajax in as much detail as possible.

第 42 道：Explain how Jsonp works (and how it's not really Ajax).

第 43 道：Have you ever used JavaScript templating, and if so, what/how?

第 44 道：Explain "hoisting".

- 第 45 道: What is FOUC? How do you avoid FOUC?
- 第 46 道: Describe event bubbling.
- 第 47 道: What's the difference between an "attribute" and a "property"?
- 第 48 道: Why is extending built in JavaScript objects not a good idea?
- 第 49 道: Why is extending built is a good idea?
- 第 50 道: Difference between document load event and document ready event?
- 第 51 道: What is the difference between `==` and `===` ?
- 第 52 道: Explain how you would get a query string parameter from the browser window's URL.
- 第 53 道: Explain the same-origin policy with regards to JavaScript.
- 第 54 道: Explain event delegation.
- 第 55 道: Describe inheritance patterns in JavaScript.
- 第 56 道: Describe a strategy for memoization in JavaScript.
- 第 57 道 : Why is it called a Ternary statement, what does the word "Ternary" indicate?
- 第 58 道: What is the arity of a function?
- jQuery-Specific Questions:**
- 第 59 道: Explain “chaining” .
- 第 60 道: What does `.end()` do?
- 第 61 道: How, and why, would you namespace a bound event handler?
- 第 62 道: What is the effects (or fx) queue?
- 第 63 道: What is the difference between `.get()` , `[]` , and `.eq()` ?
- 第 64 道: What is the difference between `.bind()` , `.live()` , and `.delegate()` ?
- 第 65 道: What is the difference between `$` and `$.fn` ? Or just what is `$.fn`.
- CSS-Specific Questions:**
- 第 66 道: Describe what a "reset" CSS file does and how it's useful.
- 第 67 道: Describe Floats and how they work.
- 第 68 道 : What are the various clearing techniques and which is appropriate for what context?
- 第 69 道 : Explain CSS sprites, and how you would implement them on a page or site.
- 第 70 道 : What are the differences between the IE box model and the W3C box model?
- 第 71 道 : What are your favourite image replacement techniques and which do you use when?

第 72 道: CSS property hacks, conditionally included .CSS files, or... something else?

第 73 道: How do you serve your pages for feature-constrained browsers?

What techniques/processes do you use?

第 74 道: What are the different ways to visually hide content (and make it available only for screenreaders)?

第 75 道: Have you ever used a grid system, and if so, what do you prefer?

第 76 道: Hav you used or implement media queries or mobile specific layouts/css?

第 77 道: Any familiarity with styling SVG?

第 78 道: How do you optimize your webpages for print?

第 79 道: What are some of the "gotchas" for writing efficient css?

第 80 道: Do you use LESS?

第 81 道: How would you implement a web design comp that uses non-standard fonts? (avoid mentioning webfonts so they can figure it out)

第 82 道: Explain how a browser determines what elements match a css selector?

Optional fun Questions:

第 83 道: What's the coolest thing you've ever coded, what are you most proud of?

第 84 道: Do you know the HTML5 gang sign?

第 85 道: Are you now, or have you ever been, on a boat.

第 86 道: Tell me your favorite parts about Firebug / Webkit Inspector.

第 87 道: Do you have any pet projects? What kind?

第 88 道: Explain the significance of "cornify".

第 89 道: On a piece of paper, write down the letters A B C D E vertically.

Now put these in descending order without writing one line of code.

Wait and see if they turn the paper upside down.

This should make the laugh and is a fine way to relieve some tension at the end of the interview.

第 90 道: Pirate or Ninja?

Bonus if it's a combo and a good reason was given (+2 for zombie monkey pirate ninjas).

If not Web Development what would you be doing?

Where in the world is Carmen Sandiego?

(hint: their answer is always wrong)

What's your favorite feature of Internet Explorer?

第 91 道：What do the terms "Filegroups", "components" and "setup types" mean in Installshield Professional?

第 92 道：Given a collection of files what process would you go through using the three items detailed in question 1 to produce an installation.

第 93 道：What do you understand by the term "Maintenance Mode" with regard to Installshield Professional?

第 94 道：What is the equivalent of a Component in Installshield Developer.

Describe an installation you have written with Installshield Professional – what did it install? Did you use customized dialogues? If so how did you do this? Did the installation require you to write your own scripted functions? If so give an example. Did you consider future upgrades of your applications – expand.

第 95 道：What do you know about Windows Installer based installations i.e. Have you had any exposure to creating them? If so expand, Where is the installation information held for such an installation?

第 19 章



人资问题

- 第 1 道：说一下你的离职原因？
- 第 2 道：上家公司的工作流程？
- 第 3 道：上家公司的前端有几个人？
- 第 4 道：毕业于哪所大学？
- 第 5 道：专业学的什么？
- 第 6 道：对这个职业有什么想法？
- 第 7 道：有什么想问的吗？
- 第 8 道：后台接触多吗？
- 第 9 道：项目怎么分配？
- 第 10 道：工作流程是怎样的？你主要负责哪块？
- 第 11 道：项目完成的进度？
- 第 12 道：平时喜欢看什么书？
- 第 13 道：和同事意见不统一怎么办？
- 第 14 道：说一下你的工作经历。
- 第 15 道：清楚地描述你的毕业时间，实习时间，工作时间。
- 第 16 道：希望处于什么样的工作环境？
- 第 17 道：上家公司在哪？
- 第 18 道：你上手一个新东西需要多久？
- 第 19 道：上家公司薪资多少，最低多少薪资？期望薪资是多少？
- 第 20 道：你懂产品吗？
- 第 21 道：上家公司的工资怎么发的？
- 第 22 道：上家公司经常加班吗？
- 第 23 道：自己对上公司的评价？
- 第 24 道：员工内部买东西有没有折扣？
- 第 25 道：你一般学习都会去什么网站？
- 第 26 道：上公司的规模有多大？
- 第 27 道：你认为凡客诚品的网站怎么样？
- 第 28 道：你认为前端工程师最注重哪一点？
- 第 29 道：你们部门是怎么分配的？
- 第 30 道：最快什么时候能入职？

- 第31道：你觉得什么公司比较适合发展？
- 第32道：要是给你时间学的话，能行吗？
- 第33道：为什么要换工作？
- 第34道：你得多久才能离职？
- 第35道：请问您的职业规划是什么？
- 第36道：你为什么选择在我们公司工作？
- 第37道：在工作的过程中，如果自己本身心情不是很好，而领导又批评了你，你会如何处理？
- 第38道：领导安排你独立完成某个项目工作的时候你是怎么想的？
- 第39道：当你遇到自己不懂或不会的问题时，你会怎么办？
- 第40道：在工作过程中，如何更好地提升工作效率？
- 第41道：你的优点是什么？
- 第42道：当你在做项目（业务）的过程中遇到新技术需要学习，你会如何处理？
- 第43道：当你接手一个需要利用新技术的项目时，你会怎么办？
- 第44道：如果上级领导对你的工作不满意，你会怎么办？
- 第45道：工作时，你认为领导要求的方式不是最好的，你会怎么做？
- 第46道：如果有合作公司老板打电话给你要你过去，待遇比现在高，你怎么办？
- 第47道：你会在什么情况下提出涨薪？
- 第48道：在工作的过程中，如果公司或者团队遇到了困境，你会怎么解决？
- 第49道：当在项目（管理）中遇到困难时，你是如何解决的？
- 第50道：你是如何看待北京的雾霾的？
- 第51道：当你面对部门不公平任务分工的时候，你会怎么办？
- 第52道：你怎么看待跳槽？
- 第53道：你只有两年工作经验，而现在应聘的是项目经理，你觉得自己可以胜任这个岗位吗？

